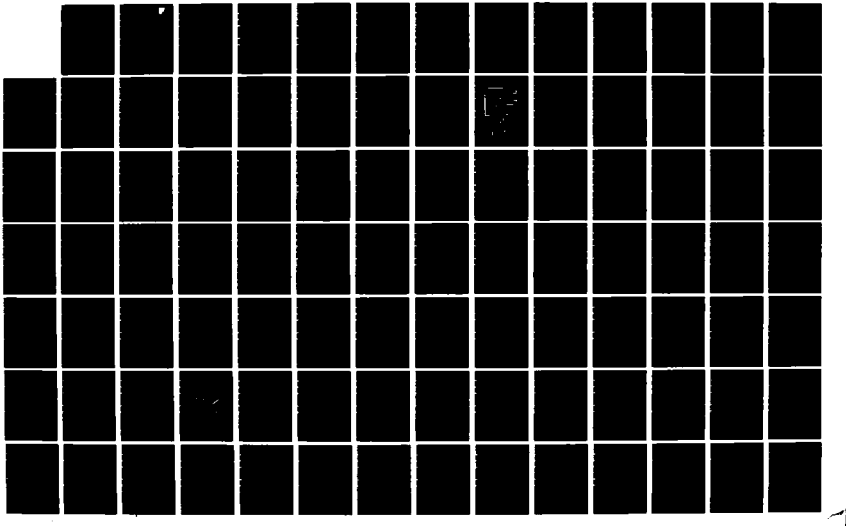
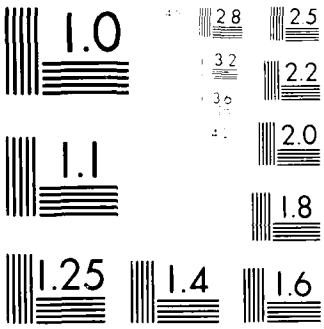
AD-A169 300 C3I (COMMAND CONTROL COMMUNICATIONS AND INTELLIGENCE) 1/3
TERADATA STUDY(U) MEASUREMENT CONCEPT CORP ROME NY
J DECKER MAR 86 RADC-TR-85-273 F30602-85-C-0029
UNCLASSIFIED F/G 17/2 NL




1.0
2.8
2.5
3.2
2.2
3.6
2.0
1.1
1.8
1.25
1.4
1.6

(12)

RADC-TR-85-273
Final Technical Report
March 1986

C³I TERADATA STUDY

Measurement Concepts Corporation

John Decker

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

This effort was funded totally by the Laboratory Directors' Fund

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-85-273 has been reviewed and is approved for publication.

APPROVED: Patricia M. Langendorf

PATRICIA M. LANGENDORF
Project Engineer

APPROVED: Walter J. Senus

WALTER J. SENUS
Technical Director
Intelligence & Reconnaissance Division

FOR THE COMMANDER: Richard W. Poulliot

RICHARD W. POULLIOT
Plans & Programs Division

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (IRDA) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

AD-A169300

ERRATA

1 August 1986

RADC-TR-85-273
C^3I TLRADATA STUDY
March 1986

Please add the following author on the cover of the report and also on the DD Form 1473.

Fred Tims

Rome Air Development Center
Air Force Systems Command
Griffiss Air Force Base NY 13441-5700

UNCLASSIFIED

REPORT DOCUMENTATION PAGE

UNCLASSIFIED	N/A
N/A	Approved for public release; distribution unlimited
N/A	
N/A	RADC-TR-[illegible]
Measurement Concepts Corporation	Rome Air Development Center [illegible]
1721 Black River Blvd Rome NY 13440	Griffiss AFB NY 13441-5700
Rome Air Development Center / [illegible]	F30602-[illegible]
Griffiss AFB NY 13441-5700	

(Include Security Classification)

[illegible] TERADATA STUDY

PERSONAL AUTHOR(S)

John Decker

13a. TYPE OF REPORT	13b. TIME COVERED		
Final	FROM [illegible] TO [illegible]	[illegible]	[illegible]

SUPPLEMENTARY NOTATION

This effort was funded totally by the Laboratory Director's Fund

COSATI CODES			SUBJECT TERMS
FIELD	GROUP	SUB-GROUP	
[illegible]	04	2.4	Database Machine
[illegible]	02	6.1	Data Management
			Relational Database

ABSTRACT (Continue on reverse if necessary and identify by block number)

The DBC/1012 is an excellent choice for new C³I [illegible] current mainframe DBMS for most C³I applications. [illegible] hour 365 day year operation using two or more host [illegible] excellent [illegible] support and good concurrency control and deadlock detection and resolution. [illegible] a very complex black box with security in hardware, [illegible] Security tools are good but probably not sufficient [illegible] significant digits floating point calculation [illegible] an excellent foreign file capability [illegible] structured queries to support [illegible]

[illegible] would be very difficult to [illegible] current C³I systems are tightly coupled.

[illegible] DBC/1012 [illegible]

DISTRIBUTION / AVAILABILITY OF ABSTRACT [illegible]

NAME OF RESPONSIBLE INDIVIDUAL

Patrick [illegible]

[illegible] and software forming a parallel Multiple Instruction Multiple Datastream (MIMD) computer. The modular design can accommodate up to 1024 processors and support a theoretical data storage capacity of one trillion bytes.

The DBC operates as a dedicated backend attached to one or more host computers. At present only IBM computers are supported. However, plans to attach to VAX have been announced. The same DBC1012 is intended to be capable of interfacing both IBM and VAX machines at the same time, thus providing an intriguing capability to access the same database using computers with truly different capabilities. Plans to interface the IBM PC have also been announced.

The DBC1012 employs a relational data model with records evenly distributed across disk units to provide parallel asynchronous processing. Access is supported by the TEradata Query Language (TEQUEL), a high-level non-procedural dialect of SQL. Modes of operation range from interactive database query in TEQUEL to invocation by application programs executing on the host. The DBC1012 should be accessible by any language resident on the host.

The system is constructed of off-the-shelf extremely reliable hardware and is assessed to be highly reliable and maintainable. It is redundant in both hardware and software with non-stop operation and non-stop repair under most failure events. The DBC1012 capacity statistics are up to 32,000 databases, 32,000 tables per database, 256 columns per table, 50,000 records (rows) per table and 30,000 bytes per field. Each disk can hold 600 megabytes plus overhead. Efforts are under way to reduce this overhead. Analysis indicates virtually no loss in parallelism for primary key activities as the system grows and the next release plans the same capability for secondary key activities.

Human factors are excellent, with particularly desirable support to data administration. It is easy to specify input validation criteria, restructure and expand machine configurations. "Create data base" privileges can be granted and controlled. Extensive reports are available covering network processor and storage utilization, user activities and privileges, and database and table ownership hierarchies.

The only identified limitations of any significance are lack of a rollforward capability announced for late 85 is software and announced without date in hardware, and lack of a capability to name indexes, which is an annoyance but should not limit usage.

EXECUTIVE SUMMARY

This effort investigated the applicability of the Teradata DBC/1012 database computer to support Command, Control, Communications and Intelligence (C3) functions.

The DBC/1012 is an excellent choice for new C3I systems. It has far more capability than current mainframe DBMSs for most C3I applications. The DBC/1012 can provide continuous 24-hour, 365-day year operation using two or more host processors. It has excellent integrity support and good concurrency control and deadlock detection/resolution. The DBC/1012 is a very complex black box with security in hardware, which makes security assessment difficult. Security tools are good but probably not sufficient for multi-compartment operation. Fifteen significant digit floating point calculations can be performed inside the database machine. The Teradata DBC/1012 has an excellent foreign file capability, good facilities to browse data, and good tools for structured queries to support looking for patterns or trends in data.

The DBC/1012 would be very difficult to embed in an existing tightly coupled information system, and most current C3I systems are tightly coupled. However, when additional functions must be added to current systems, implementing them on a Teradata 1012 might be desirable.

The Teradata DBC/1012 is a unique architecture of multiple processors, direct access st 'age devices and software forming a parallel Multiple Instruction Multiple Datastream (MIMD) computer. The modular design can accommodate up to 1024 processors and support a theoretical data storage capacity of one trillion bytes. The largest configuration currently built consists of 60 processors and 60 disks.

The DBC/1012 operates as a dedicated back end attached to one or more host computers. At present, only IBM and IBM plug-compatible computers are supported. However, plans to attach to other manufacturers' machines are in progress. Interfaces to the IBM PC are also in the planning stages. The same DBC/1012 is intended to be capable of interfacing machines of different manufactures at the same time, thus providing an intriguing capability to access the same database using computers with truly different capabilities.

The DBC/1012 employs a relational data model with records evenly distributed across disk units to provide parallel asynchronous processing. Access is supported by the TEradata QUEry Language (TEQUEL) - a high-level non-procedural dialect of SQL. Plans have been announced to be fully compatible with SQL when the standard is approved. Modes of operation range from interactive database query in TEQUEL to invocation by applications programs executing on the host. COBOL and PL/1 preprocessors are available at present. The DBC/1012 should be accessible by any language resident on the host.

The system is constructed of off-the-shelf extremely reliable hardware, and is assessed to be highly reliable and maintainable. It is redundant in both hardware and software with non-stop operation and non-stop repair under most failure events. Attempts to devise a problem to saturate the only non-multiply redundant component of the DBC/1012, the Y-net, were finally successful in a 128 processor, 498 terminal system simulation where each terminal was updating the entire screen from the database, pixel by pixel. No problem at all likely in practice achieved 10% usage of the Y-net.

The DBC/1012 capacity statistics are up to 32,000 databases, 32,000 tables per database, 256 columns per table, 30,000 records (rows) per table, and 30,000 bytes per field. At present, each disk can hold 300 megabytes plus overhead. Efforts are under way to reduce this overhead. Analysis

indicates virtually no loss in parallelism for primary key activities as the system grows and the next database software release plans the same capability for secondary key activities.

Human factors are excellent, with particularly desirable suppport to data administration. It is easy to specify input validation criteria, restructure and expand machine configuration. 'Create data base' privileges can be granted and controlled. Extensive reports are available covering network processor and storage utilization, user activities and privileges, and database and table ownership hierarchies.

The only identified limitations of any significance are lack of a roll forward capability, which is announced for late 1985 in software and announced without date in hardware, and lack of a capability to name group indexes, which is an annoyance but should not limit usage.

The analysis consisted of a study of the DBC/1012 characteristics in terms of functionality, throughput, data access, response time, fault tolerance expandability, error recovery, bundled software and existing interfaces. The study included literature analysis including patents, performance modeling, extensive discussion with Teradata corporate personnel, and limited hands on experience.

The most significant aspect of the DBC/1012 architecture is the method of integrating the system components under the Y-net interconnection bus. The components are off-the-shelf devices, currently Intel 8086 processors, which can be upgraded when newer components become desireable. The inter-connection scheme allows the configuration components to run asynchronously, passing and receiving messages when necessary to other system components. Messages are passed within the system under a protocol which uses codes embedded in each message to determine message priorities. Thus, no controlling processor is needed to govern the parallel processing network.

The DBC/1012 currently has a small customer base. Eight customers were contacted, three of which were willing to provide detailed information on their specific DBC/1012 configurations and assessment of the product. In general, the customers were pleased with the system itself, and with the hardware and software support provided by Teradata Field and Systems Engineers. Specific negative comments were directed toward small errors in software which either were repaired in subsequent releases or are being addressed. The general consensus is that the hardware and software are robust and mature to a degree unusual in first offerings.

The hands-on session appeared user-friendly and demonstrated the ease with which database requests would be generated by a user with a general working knowledge of SQL.

A microprocessor-based analytic queueing model was built to predict performance characteristics. The results of this model closely matched those obtained during Teradata benchmark tests. Parameters of the model enable specification of configuration in terms of number of and mix of devices; devices in terms of processor rates and system buffer size, load in terms of requests per user per minute and number of users, data base characteristics and application parameters.

Scenarios were developed to examine the effects on the system of specific types of requests to investigate typical photo interpretation and indications and warnings activities, and to duplicate the benchmarks to validate the model and extrapolate to larger configurations.

For primary key retrievals, a linear performance curve was obtained, with absolute processing rates theoretically reaching almost 5000 transactions per second for an optimally specified configuration of 1024 processors. For secondary retrievals, the performance curve leveled off at 12 transactions per second. Modeling the announced new software release yielded secondary key operations in line with primary key retrieval.

Update operations were also linear. The maximum configuration supported 720 transactions per second. Joins peak at approximately 100 transactions per second, and performance levels off at 256 processors, due to excessive internal communications.

The PI scenario yielded 1100 transactions per second for an optimally configured 1024 processor system. Performance improvement was linear under expansion of configuration and workload. The I&W scenario exibited leveling off due to Join operations. Maximum performance under optimal configuration reached 475 transactions per second.

TABLE OF CONTENTS

LIST OF APPENDICES

LIST OF FIGURES

LIST OF TABLES

SECTION 1
GENERAL

1.1 Introduction

This is the Final Technical Report under the C^3I Teradata Study effort, Contract Number F30602-85-C-0029. It contains a summary of those analysis activities described in two previously delivered documents: Teradata DBC/1012 Data Base Machine Capabilities and Characteristics Report and Analysis of the Teradata DBC/1012 Data Base Machine's Applicability to C3I Operations. A detailed description of the DBC/1012 performance analysis activities and tradeoffs and technical considerations is also presented to develop an overall assessment of the support potential the DBC/1012 holds for the C^3I operations environment.

1.2 Project References

The following documents, in addition to those reports specified above, are directly applicable to the completion of the project:

a. Statement of Work (SOW), C^3I Teradata Study, RADC PR No. I-5-4338, July 5, 1984.

b. C^3I Teradata Study Technical Proposal, Mc^2, October 11, 1984.

A reference list of material consulted during the performance of this project is provided in Appendix A of this report.

1.3 Terms and Abbreviations

Terms and acronyms used in this document and subject to interpretation by the reader are listed in Appendix B.

1.4 Report Organization

The remaining sections of this report are as follows:

SECTION	TOPIC
2	Project flow, task descriptions, and technical approaches employed.
3	General description of the DBC/1012's capacities, configuration, the TEQUEL DBMS language, and host system requirements.
4	Engineering assessment of the DBC/1012: analysis of Teradata U.S. Patent information, benchmark testing, hands-on experience, customer surveys, and reliability analysis.
5	Performance modeling: the Mc^2 performance model, model parameters, test descriptions, results, and performance assessment.
6	Summary of DBC/1012 applicability to C^3I operations in the areas of functionality, integrity, data base administration, and insertion into existing systems.
7	Overall assessment summary of the DBC/1012 and recommendations.

SECTION 2
PROJECT SUMMARY AND TECHNICAL APPROACH

Section 2 defines the objective of the C^3I Teradata Study effort, details its associated tasks, and describes the technical approach followed in conducting the effort.

2.1 Objective

The objective of this effort was to investigate the applicability of the Teradata DBC/1012 data base machine to Command, Control, and Communications (C^3I) operations. The context of this investigation included the study of the DBC/1012's architecture and functionality and the assessment of how these features mapped to the C^3I data management domain.

The DBC/1012 is a distributed, flexible system configuration which offers parallel processing support to DBMS operations. Before the advent of the DBC/1012, commercially available data base machines were basically sequential, "back-end" processors connected to a host computer via a data channel. Although some of these machines have incorporated special purpose function architecture (SPFA) for the optimization of data access operations, they have not exploited the advantages of parallelism. Parallel-architecture data base machines have been the subject of concerted research and development efforts, but they exist as one-of-a-kind or "paper" systems only. Thus the DBC/1012 occupies a unique position in the data base processing environment: it is both a parallel multiple-instruction/multiple-datastream (MIMD) architecture, and it has been employed in the applications environment by various commercial organizations since early 1984.

Automated C^3I systems have grown from the simple weapon and target inventories of the 1950s, containing a few thousand records, to the on-line, near real time, multi-user systems of today, containing hundreds

of thousands of records, and performing many of the functions previously performed by humans, or not performed at all. Advancing technology has made it possible to address requirements which have existed all along, but have previously been impossible to satisfy. This increase in capabilities has involved advances in both computer systems technology (particularly in storage technology and information sciences), and military science (particularly in force management and targeting concepts). The advances in estimative intelligence methodology, which attempts to discern the meaning or significance of reported events, attempts to infer additional facts, and attempts to predict future events; and the phenomenal growth in sensor technology have also contributed to the growth of C^3I systems. In addition, improvements in communications have made it possible to manage forces in ever smaller increments.

Corresponding to the growth of C^3I systems and capabilities has been the increase in the magnitude and complexity of the requirements levied upon them. As the threat has increased, it has been incumbent upon C^3I systems to do more things, using larger data bases, and to do it all faster. In addition, the consequences of not doing it right have become more severe. Figure 2-1 lists these increasing demands on C^3I systems.

In summary, the objective of ths effort implies several questions:

- Does insertion of a DBC/1012 into a C^3I system enhance operations in the areas of functionality, integrity, data base administration, and performance?

- Are there classes of C^3I systems (based on data base size, operational performance requirements, specific functionality, etc.) which would particularly benefit from insertion of a DBC/1012?

MUCH LARGER DATA BASES

- o MULTI-SENSOR/MULTI-SOURCE CORRELATION
- o EXPANDED COVERAGE
- o MORE COMPREHENSIVE, ON-LINE HISTORICAL DATA

HEAVIER TRAFFIC

- o MORE USERS
- o HEAVIER DEMAND PER USER
  - - ENHANCED INTELLIGENCE ANALYST AIDS
    - -- ENHANCED CORRELATION FUNCTIONS
    - -- ENHANCED GRAPHICS SUPPORT
      - --- BAR CHARTS
      - --- CURVES
      - --- COLOR
      - --- MAP OVERLAYS
      - --- 4D
- o DRAMATICALLY HIGHER UPDATE RATES

REQUIREMENT FOR FASTER RESPONSES

FASTER MOVING EVENTS BEING TRACKED

FIGURE 2-1 INCREASED REQUIREMENTS

- Are there situations which cause significant degradation of parallelism in the DBC/1012, and if so, are they manifested in any phases of C^3I processing?

2.2 Task Descriptions and Technical Approach

The C^3I Teradata Study consisted of four tasks:

- Task 1: Determine and document the characteristics of the DBC/1012 with respect to functionality, throughput, upgradability, data access, response time, fault tolerance, extensibility, error recovery, bundled software, and existing interfaces.

- Task 2: Determine and document the applicability of the DBC/1012 to C^3I operations in terms of current capabilities, identified needs, and future trends.

- Task 3: Identify how the introduction of a DBC/1012 can enhance the data administration functions of the overall data handling system.

- Task 4: Analyze and document tradeoffs and technical considerations resulting from performance of the above three analysis tasks to provide a firm foundation regarding integration of the DBC/1012 in a C^3I environment.

The technical approach employed in performing these tasks consisted of three main activities which spanned all tasks of the effort.

The first activity was a data collection and analysis of Teradata system manuals, data base technology literature, and a review of C3I systems documentation. The goals of this activity were to become familiar with all aspects of the DBC/1012 hardware, software, and data base management, and to establish a foundation of general C3I requirements against which the perceived capabilities and/or shortcomings of the DBC/1012 could be evaluated.

The second activity was an engineering analysis of the DBC/1012. An analysis of the architecture of the machine was peformed through the study of the United States Patents issued for the eventual DBC/1012 parallel configuration and interconnection network. Data collection activities were conducted at the Teradata corporate headquarters and manufacturing facility to ascertain DBC/1012 future product directions, and to resolve issues raised by the Project Team in the initial phases of DBC/1012 study. Survey forms were developed and sent to Teradata customers in attempt to assess reaction to the DBC/1012 by its user community. Benchmark performance data was collected and analyzed to determine achievable processing rates and factors which affect processing performance. Limited hands-on experience was conducted to determine the user-friendliness of the system and witness first-hand response rates for differing functions. A reliability analysis was also conducted. Component failure data was collected and a Failure Mode Effect Analysis (FMEA) was developed to indicate potential points of failure in the configuration and their effects on total system operation.

Since the contract did not include access to a DBC/1012 site, and since most of Teradata's performance benchmark materials involved sensitive customer or corporate confidential data, a performance model was constructed on an Mc2 facility microcomputer. The model was built in much the same way as the performance predictive model created by Teradata personnel; data base requests were subdivided into machine DBC/1012 processor component operations, and ultimately divided into cycles whose times were based on the hardware characteristics of each component. The

performance model facilitated the analysis of various data base requests under changing parameters, such as number of users, variations in the configuration of the DBC/1012, and differing data base element sizes (e.g.; row, column, etc.). The performance model was also used to predict response times for varying C^3I workloads which were postulated based on requirements specified in the project references and known from past corporate C^3I studies. Model validation procedures where followed to the extent that performance characteristics derived from model runs were similar to those collected from benchmark testing. For example, where the Teradata Benchmarks showed linearity of performance, the model also produced these results. The model was not meant to precisely predict a processing rate under given conditions; rather it was developed to be an aid in analyzing DBC/1012 performance behavior for currently non-existant high-end systems. In many cases, model outputs were very close to actual performance figures.

Figure 2-2 illustrates the general task flow of the effort, highlights the technical approach employed, and summarizes the effort's resultant products.

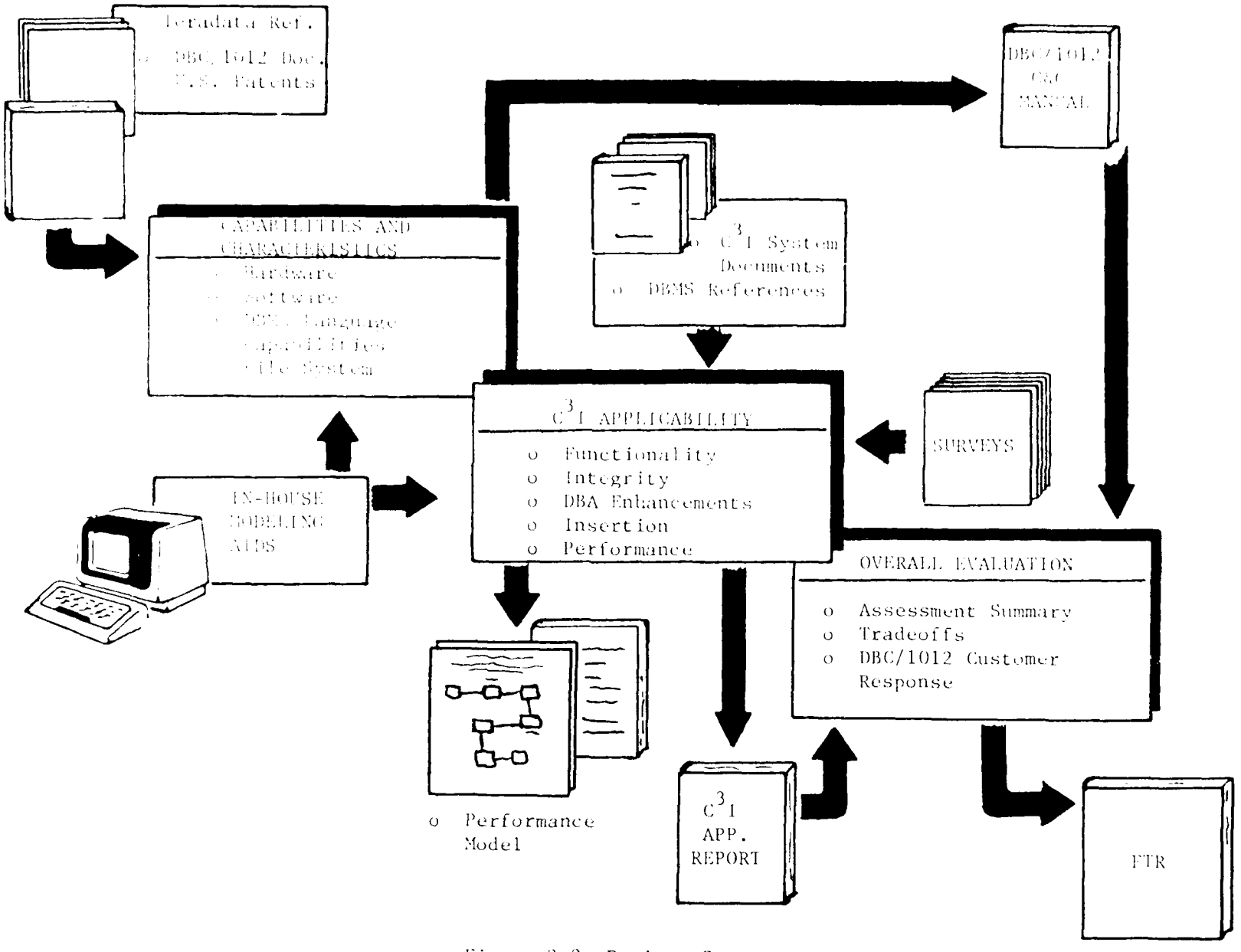


Figure 2-2 Project Summary

SECTION 3
DBC/1012 DESCRIPTION

The DBC/1012 is essentially a standalone data base management system which includes host-resident software supplied by Teradata. The fully redundant hardware, firmware, software, and an optional "fallback copy" of user data, resides within the DBC/1012 system which attaches to one or more host CPUs.

The DBC/1012 employs the relational model of data as opposed to a hierarchical or network structure. Records are evenly distributed across a number of processors and disk storage units allowing for asynchronous processing of data. The modular design of the DBC/1012 Data Base Computer can accommodate up to 1,024 processors, theoretically allowing for up to a terabyte (1,000,000,000,000) of data.

Access of the data takes place through a high-level nonprocedural language called the TEradata QUEry Language (TEQUEL). TEQUEL statements are used for data definition, data manipulation, query, report writing, and control.

A Data Dictionary/Directory contains information which may be accessed by the user. Information on tables, views, macros, data bases, users, ownerships, space allocation, accounting, and access rights. Information in the Data Dictionary/Directory is updated automatically during the processing of TEQUEL data definition statements, and used by the TEQUEL Parser to obtain information required to process all TEQUEL statements.

Additional software supplied by Teradata includes general purpose utilities for dumping and restoring data bases for bulk loading of data from the host computer. System maintenance facilities provide a trouble log, automatic testing of processors at startup, and isolation, diagnosis, and repair of offline processors during online operation.

This section describes the DBC/1012 configuration, the TEQUEL DBMS language, configuration and data base capacities, and the host system requirements for integrating a DBC/1012 into an existing ADP environment.

3.1 DBC/1012 Configuration

The following paragraphs present the hardware and software components found in a DBC/1012 configuration. Figure 3-1 illustrates the DBC/1012's primary components.

The major components of the DBC/1012 are:

- Host-Resident Software
- Interface Processors (IFPs)
- Ynet Interconnection Network (Ynet A and Ynet B)
- Access Module Processors (AMPs)
- Disk Storage Units (DSUs)
- System Console and Printer

3.1.1 Host-Resident Software

The function of the Host-Resident Software is to allow users to conduct sessions with the DBC/1012 from the Host CPU in interactive, batch, or on-line environments. CLI service routines, PL/1 and COBOL preprocessors, Interactive TEQUEL, and Batch TEQUEL are the DBC/1012 interface vehicles.

Host-Resident Software supplied by Teradata comprises of the following:

- Teradata Director Program (TDP)
- Call-Level Interface routines (CLI)
- COBOL Preprocessor
- PL/1 Preprocessor
- Interactive TEQUEL (ITEQ)
- Batch TEQUEL (BTEQ)

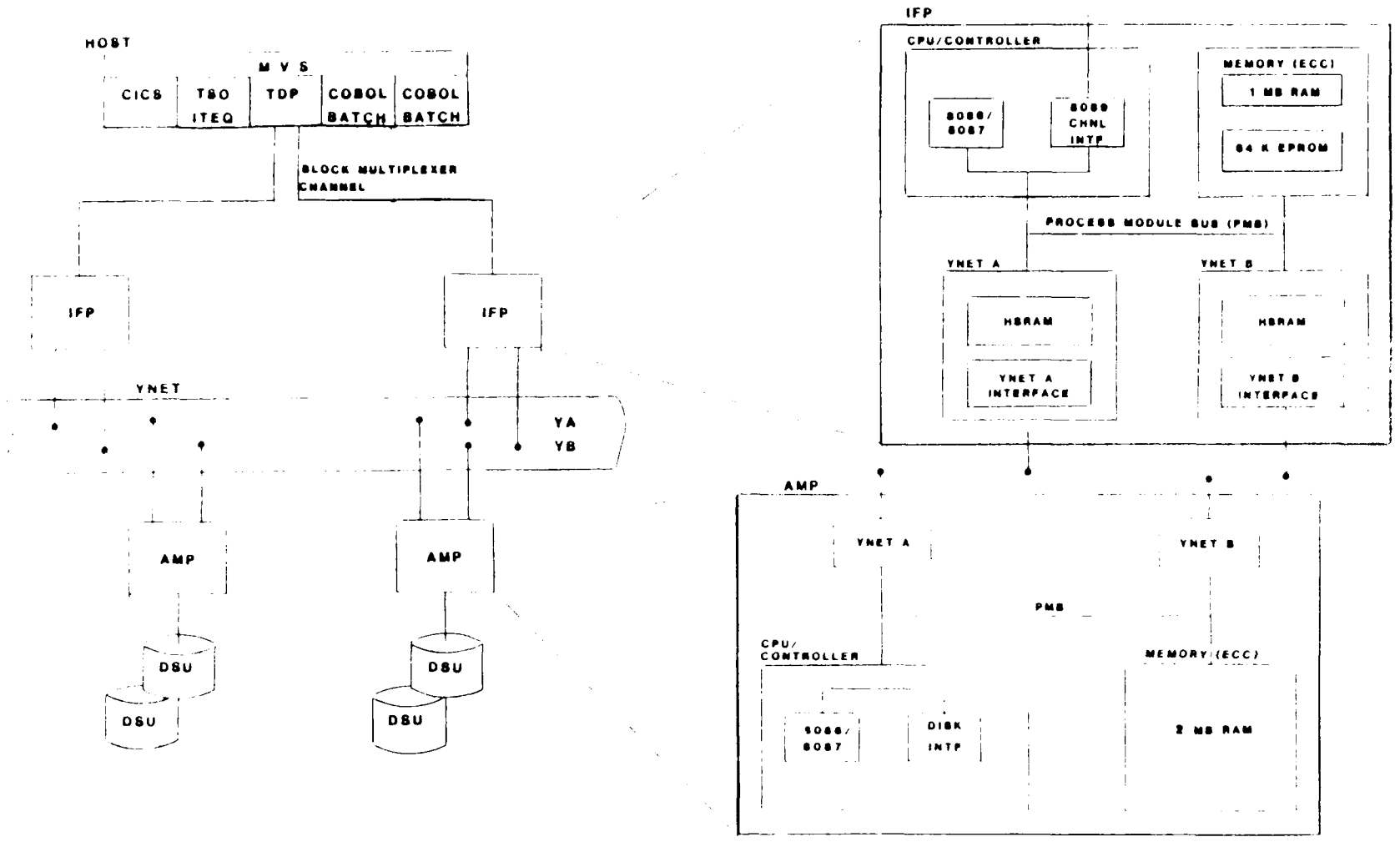


Figure 3-1 **DBC/1012 CONFIGURATION**

The Teradata Director Program (TDP) manages communication between an on-line transaction program, a batch program, or an ITEQ/BTEQ session and the DBC/1012. The TDP runs in a dedicated address space in the host CPU. CLI service routines generate supervisor calls (SVCs) from TEQUEL requests. The TDP uses the SVCs to create request messages which are then communicated over a block multiplexer channel for processing by an IFP.

Once the request has been processed, the response is transmitted back to the TDP via the multiplexer channel from the IFP and returned to the proper program or session via a CLI service routine.

Call-Level Interface (CLI) service routines, provided in a library by Teradata, may be used by an application program written in a high level language that has a CALL statement. CLI service routines handle the actual program interface with the DBC/1012 Data Base Computer. CLI routines also maintain a context for each session and each request thereby providing session control. A list of all the CLI routines is provided in the DBC/1012 Data Base Computer Host Interface Manual.

The COBOL and PL/1 Preprocessors convert TEQUEL statements embedded in COBOL and PL/1 programs into calls to CLI routines and Preprocessor support routines. The Preprocessors permit the full usage of TEQUEL data manipulation capabilities by allowing TEQUEL statements within COBOL and PL/1 source programs.

Interactive TEQUEL (ITEQ) enables a user at the terminal to interact directly with the DBC/1012 without the need to develop application software [DBC-1:3-19]. ITEQ includes functions for controlling the operation of the terminal, entering, editing, and executing TEQUEL statements, formatting output, writing reports, and executing macros.

The Batch TEQUEL facility (BTEQ) allows a user to execute, in batch, a script that consists of a number of TEQUEL statements. It also provides functions for reading and writing to host files, using more than one session for a job, formatting output, and writing reports.

3.1.2 Interface Processor

IFPs manage the dialog between a user session in a host computer and the DBC/1012. IFPs parse, translate and analyze a data base request from the host CPU, and decode the request message into a series of request steps which are then transmitted over the Ynet to the AMPs. IFPs also coordinate responses as they are returned back over the Ynet from the AMPs.

The IFPs hardware configuration is composed of a CPU/Controller, memory, and two Ynet Interfaces. The CPU/Controller is based on the Intel 8086 and 8087 processors (soon to be upgraded to 80286/80287 processors), and an 8089 input/output processor for managing the interface to the host multiplexor channel. The IFP memory consists of two megabytes of error checking, single bit correction RAM, and 64K bytes of EPROM. The two Ynet interfaces each contain 32K bytes of high speed RAM used to buffer message blocks. The CPU/Controller, memory, and the two Ynet Interfaces are implemented on four separate circuit boards.

Residing in the IFP is the DBC/1012 software consisting of the Teradata Operating System (TOS), the TEQUEL System, and parts of the Data Base System.

An IFP has five basic components:

- Session Control
- Host Interface
- Tequel Parser
- Dispatcher
- Ynet Interface

Session Control processes logon and logoff requests from the host and establishes sessions.

The Host Interface controls the exchange of messages between the TDP and the DBC/1012.

The TEQUEL Parser interprets a request message from the TDP, checks it for syntax and evaluates it semantically. Next, it decodes the request into a series of data manipulation steps necessary for routing the request to the appropriate AMP(s), and passes the steps on to the Dispatcher.

The Dispatcher controls the sequence in which the data manipulation steps are executed. These steps are then passed to one of the two available Ynet interfaces based on a Least-Recently-Used algorithm.

The Ynet interfaces control the transmission of the messages (data manipulation request steps from the IFP to the AMPs, and results/status/error responses from the AMPs to the IFP). Responses (results or error messages) are subsequently returned to the host-resident TDP from the IFP over the host DBC/1012 block multiplexer channel.

3.1.3 Ynet

The Ynet functions as the interconnection network between the IFPs and the AMPs using an array of logic containing sort and merge functions. The DBC/1012 is equipped with two Ynet networks; Ynet-A and Ynet-B. Both networks operate concurrently sharing the communication of request steps from the IFPs to the AMPs and returning data from the AMPs back to the IFPs. Each Ynet serves as a backup of the other. They are independent of each other in physical partitioning, electrical power, internal clocks, and interface logic.

The Ynet node board contains logic for up to eight processor modules. A node expansion board contains logic to connect up to four node boards or four node expansion boards in order to extend the Ynet.

The Ynet interconnection network is a tree structure which employs clocks to strobe data/status messages from one level to another in the tree hierarchy. At the bottom of the hierarchy are DBC/1012 processors; the IFPs and AMPs. Message output from these processors must compete with each other at each tree node on their journey to the tree apex according to the following message priority scheme (listed here in descending order):

- o First arrival at a network node
- o Lowest command code
- o Lowest key field
- o Shortest key field
- o Lowest data field
- o Shortest data field.

When a message gains control of a node, it may then compete for control of the next higher node. Losers are terminated; their originating processors receive a "collision" indication. The processors must then wait until the winning message has been transmitted before they can resubmit their messages to the network (a winning message is a message which has arrived at the apex node).

Since fault detection is vital to the successful operation of the DBC/1012 parallel configuration, error messages are given the lowest value command codes. Thus error notification messages, such as processor faults, always preempt normal data request and response message traffic.

Once a message reaches the apex node of the network, it no longer is subject to contention and is broadcast to all processors on the network.

The number of node levels or tiers in a network of processors can be determined by the following formula:

$$N=Int(\log_2 P)$$

where

N = number of node levels

P = number of IFPs and AMPs

Int is the Integer function

Thus, for a maximum configuration (see Section 3.3) of 1024 processors, there are ten tiers in the interconnection tree.

3.1.4 Access Module Processor

The AMPs control access to the Disk Storage Units (DSUs). They receive the requests forwarded by the IFPs, perform the required data manipulation, and send the appropriate response back to the IFPs over the Ynet. AMPs also may communicate with each other; for example, during the execution of JOIN operations, and in deadlock detection/resolution processes.

Each AMP can currently control access for up to two DSUs. Future enhancements may expand AMP control to four DSUs.

AMPs can be "clustered" by the system administrator to provide additional data availability. Clustering increases the probability that all system data is available even if two or more AMPs fail simultaneously, as long as the AMPs are not within the same cluster.

The hardware for the AMP is similar to that of the IFP. The AMP contains two Ynet Interface Modules, Memory, and a CPU/Controller. The Controller in the AMP differs in that it contains a Signetics 8X300 microprocessor for managing the industry standard SMD interface to the Disk Storage Unit, rather than the Intel 8089 processor for I/O control.

Residing within each AMP are the Teradata Operating System and the Data Base System. The Data Base Manager, DBM, is a subsystem of the Data Base System.

The primary functions of the DBM subsystem are:

- o Receiving steps from the IFP
- o Processing the steps against the data base
- o Logging
- o Sending update messages to fallback AMPs
- o Data base reorganization
- o Returning responses back to the Dispatcher in the IFP

The AMPs also provide for the transition betwen logical organization of data and the physical organization of data on the DSUs.

3.1.5 Disk Storage Unit

DSUs are currently Fujitsu M2351 Winchester-type disk drives. Each has a 474 megabyte capacity. They are fixed, sealed modules with an average seek time of 18 milliseconds and a transfer rate of 1.9 megabytes per second. After formatting by Teradata into:

- o 842 cylinders per disk
- o 20 tracks per cylinder
- o 46 sectors per track
- o 512 bytes per sector

approximately 300 megabytes are available for user tables. More DSU capacity is planned. Newer DBC/1012 configurations will be furnished with CDC 515 megabyte DSUs.

If two DSUs are used per AMP, available storage is increased by a factor of 2.2, since the DSU pair is considered as one logical disk. Portions of the DSU system area are not replicated on the second disk.

Each DSU has a System Area and a Table Area. The System Area contains system programs and system tables. The Table Area is made up of a Prime Copy Area where user data base(s) reside. Upon request, a Fallback Copy Area and additional Spool File Areas may also be allocated within the Table Area.

When first installed in an organization, the DBC/1012's DSUs contain a single data base named "DBC". It is from this space that the system administrator assigns space to all other organizations.

Residing on each cylinder of a DSU is a Cylinder Index to facilitate the retrieval of rows and to locate the physical storage of data on the disks. The Cylinder Index contains a Data Block Descriptor List (DBList). An entry in the DBList specifies a table identifier of the rows stored in a block, the row identifier of the first row in the block, and the disk address of the block. The DBList is sorted in ascending order of table identifier and row identifier of the first row in each block so that a binary search can efficiently locate the block where a specific row is stored [DBC-2:5-9].

3.1.6 System Console and Printer

The System Console, an IBM PC, allows the user to communicate with the DBC/1012 in order to display reports on system status, current configuration, and performance, as well as to control system and diagnostic operations. A hard copy of the reports may be obtained on the optional System Printer.

Through any connected processor, an operator can communicate with all the other processors in the system via the Ynet. Certain system functions, such as the rebuild and the reconfiguration may also be invoked from the console. Offline utility and diagnostic programs are also available on diskette and can be loaded and executed on a processor under the control of the console.

3.2 System Flow

The System Flow between Host CPU and DBC/1012 can be described in general terms as follows:

A user application generates a TEQUEL request. The request is transmitted by the Teradata Director Program (TDP) over the host block multiplexer channel to an Interface Processor (IFP) of the DBC/1012. The IFP decodes the request into a series of steps and routes them over the Ynet to the appropriate AMP(s) and waits for a response request to be returned from the AMP(s). The IFP then returns the response request to the appropriate user application. A more detailed explanation follows.

Through a COBOL, PL/1, ITEQ, or BTEQ TEQUEL statement, or a high-level language's CALL statement to a CLI service routine, a TEQUEL Request is generated. The TEQUEL Request is announced to the Teradata Director Program via the Teradata supplied supervisor call (SVC), or by cross-memory services, depending on the version of host-resident software being used. From the TEQUEL Request, the TDP creates a Request Message.

A message is made up of a message header, a request number, and message data. The message header contains a session number. Message requests also specify whether response data is to be returned in record or field mode. Parcels are the logical subdivisions of a message. Physical subdivisions of the message, Packets, are transmitted across the host multiplexer channel to the IFP.

The IFP's Host Interface receives the message from the TDP across the block multiplexer. The Session Control within the IFP establishes a session and request number as well as establisning a host ID number.

The TEQUEL Parser, within the IFP, interprets the message, applies syntax checks, evaluates it semantically, refers to the Data Dictionary/ Directory, to resolve symbolic data names and to make integrity checks, and finally decodes the request message into a series of low-level data manipulation steps necessary for routing and resolving the request.

Request steps are then passed to the Dispatcher wnich contains execution and response control. Once the sequence in which the steps are to be executed is determined, the steps are passed to the Ynet interface. Depending on whether or not the request is a prime-key request or a multiple row request, the IFP transmits the step over the Ynet to a specific AMP or all to AMPs.

The request steps are accepted by the AMP's Ynet Interface and acted upon by the Data Base Manager Subsystem to perform the indicated functions. The AMP prepares a response request and transmits it back over the Ynet to the IFP Dispatcher.

The IFP Dispatcher's response control is responsible for sending the response message back to the user application input buffer.

3.3 The TEQUEL Language

TEQUEL is a non-procedural data base language that provides facilities for data definition, query/update, and administration. TEQUEL is, in many respects, similar to IBM's SQL dialect of SEQUEL. Teradata states that the TEQUEL Data Manipulation (DML) is approximately 80% compatible with SQL DML. Ongoing TEQUEL DML enhancements are closing the gap.

The following paragraphs discuss the three general classes of TEQUEL statements and overview similarities and differences between TEQUEL and SQL. Detailed statement descriptions are contained in Appendix C of this report.

3.3.1 Data Definition Language (DDL)

The data definition conventions supported by TEQUEL conform to the relational model of data, that is, data is defined as consisting of tables of information (relations) in a flat structure (i.e., no repeating groups, no "owners"). The terminology maps into relational and classical terminologies as follows:

TEQUEL	RELATIONAL	CLASSICAL
DATA BASE	DATA BASE	DATA BASE
TABLE	RELATION	FILE
COLUMN	DOMAIN	FIELD
COLUMN NAME	ATTRIBUTE	FIELD NAME
ROW	N-TUPLE	RECORD
FIELD, VALUE	ATOM, TUPLE	DATA ELEMENT

3.3.2 Data Manipulation Language (DML)

TEQUEL performs the relational operations of PROJECT, SELECT, and JOIN. These operations are not explicitly invoked by name, but are implicit in TEQUEL statements and clauses such as RETRIEVE FOR EACH and CREATE VIEW.

3.3.3 Data Administration Language (DAL)

DAL commands are used by the Data Base Administrator to tune the system, control access to data, allocate resources to users, and monitor use.

Some DAL commands have data definition and data manipulation characteristics (such as CREATE DATABASE), but are included in this category because they are not normally invoked by end users or programmers.

3.4 Capacities

3.4.1 Configurations

The minimum configuration as stated by Teradata is two IFPs, four AMPs, one double Ynet, and four DSUs. This configuration requires two hardware cabinets.

Teradata states that the maximum DBC/1012 configuration is 1024 processors. These processors include both IFPs and AMPs. Currently, two DSUs can be attached to an AMP. The Teradata briefing handout given at the 1985 Syracuse University Minnowbrook Workshop on Data Base Machines showed four DSUs attached to a new 80286-based AMP. A four DSU/AMP capability has not yet been established. Teradata states that this capability will be influenced by currently ongoing performance studies using the upgraded Intel processor.

The number of IFPs in a configuration must be at least two and can never be greater than the number of AMPs. The number of IFPs installed in a configuration depends on the forecasted transaction workload. More concurrent users or user-generated sessions require more IFPs to convert data base requests into process steps. The AMPs perform the actual data base processing in a configuration. Thus, size of the data base and complexity of data base operations are used in determining the optimum number of AMPs for a given configuration.

3.4.. Data Base Capacities

During the first phase of the effort, a model was constructed to calculate maximum storage capacity of the DBC/1012 based on 16 AMPs per IFP and two DSUs per AMP using four sets of assumptions concerning configuration and application. Results were as follows:

ASSUMPTION SET	DBC/1012 CAPACITY
The Theoretical Maximum	.636 Terabytes(TB)
A "representative" C^3I record structure	.441 TB
A "representative" Cartographic record structure	.545 TB
Systems Manual [DBC-3:6-7 to 6-10] example	.174 TB

If the FALLBACK option is exercised, these capacities are divided by two. Differences in capacities can be attributed to the overhead involved in describing row fields and indexes. More fields in a row result in more header information being maintained.

The above capacities were based on the 474MB DSU and a row format which will shortly be replaced. Total storage capacity on the DBC/1012 will be expanded based on:

- o 515MB DSUs on newer configurations;
- o Four DSUs available per AMP; and
- o A new row format specification which is estimated by Teradata to decrease storage overhead by 40%.

Other capacity statistics specified by Teradata are as follows:

- Maximum Number of Data Bases 32,000
- Maximum Number of Tables per Data Base 32,000
- Maximum Number of Columns per Table 256
- Maximum Record (Row) Size 30,000
- Maximum Field Size 30,000

3.5 Host System Requirements

Host system requirements for DBC/1012 installation are as follows:

- Host CPUs supported:
  - IBM 370 Models 148, 155 with DAT, 158, 168, 3031, 3033, 308x, 309x, 43xx on any UP, AP, or MP configuration;
  - Plug Compatible Machines such as:
    - -- NASCO
    - -- Amdahl
    - -- Magnuson
    - -- IPL
    - -- etc.
- Host Operating Systems Supported:
  - OS/VS2-MVS Release 3.8 and above or MVS/SP Release 1 or 3
    - -- SP 1.3 Required for CICS support
  - VM/CMS

o Peripheral Requirements

- Block Multiplexer Channel
- 3270 type terminals

Multiple host processors can share a DBC/1012. For each host processor, Teradata recommends that a minimum of two IFPs should be used to provide adequate redundancy in controlling user sessions.

Discussions with Teradata personnel indicate that many potential customers expressed a requirement for DBC/1012 compatibility with DEC VAX systems. Plans for such an interface are very preliminary at this time. During discussions at the Syracuse University Minnowbrook Workshop on Data Base Machines, it was revealed that a PC/SQL interface was being planned for the DBC/1012, which implies that direct IBM PC - DBC/1012 interfacing will be possible. Information on when this capability will be available is currently unavailable.

Teradata's initial strategy in marketing the DBC/1012 was to bring a full-function, high performance product to the IBM commercial data base processing environment. Product enhancements, such as DBC/1012 processor upgrades and DBMS performance improvements have been made on a continuing basis since the machine's inception. These enhancements are, no doubt, based on experiences gained from customer installations and continual experiments conducted in its Benchmark Laboratory. Teradata now intends to expand its customer base by offering DBC/1012 compatibility to other host computers. More detailed announcements can be expected in the future.

SECTION 4

ENGINEERING ASSESSMENT

Our assessment of the [illegible] has been formulated, in part by studying the design basis of the machine; i.e., the United States Patents, collecting and reviewing benchmark performance tests, soliciting and analyzing customer reactions, limited hands-on experience, and performing a reliability-maintainability study. These activities are based on the machine as it exists on the design boards and in the field, as opposed to machine model simulation analysis which is discussed in the next section of this report.

4.1 Patent Study

Two patents [illegible] the [illegible] design:

- [illegible]; "MULTIPROCESSOR INTERCOMMUNICATION [illegible]", [illegible].
- [illegible]; "DATA PROCESSING SYSTEMS AND [illegible]", [illegible].

Both patents were filed on [illegible] April 1981. Patent No. 4,445,171 describes the concept of an interconnection network which features bidirectional message paths and active logic. This patent forms the basis for the Ynet. Patent No. 4,[illegible] refers to the ways that the active logic interconnection network can satisfy the control and data processing requirements found in multiprocessor configurations. These requirements are briefly stated as follows:

- Single points of control/interfacing to the configuration;
- Processor fault [illegible] and graceful degradation/recovery;

- Resolution of resource contention by competing processes; and

- Distribution of tasks/information to configuration components resulting in optimum use of all processors in the configuration.

Section 3 provided a detailed explanation of the active logic interconnection network, implemented as the Ynet. The following subsections describe what we consider as key features of the patent design and how these features satisfy the above-stated requirements.

4.1.1 Key Features Found in the Patents

The first feature of the interconnection network is its binary tree nature. As discussed in Section 3, the number of network tiers (levels) only increases by one each time the number of processors (AMPs, IFPs) is doubled. Thus, the time for messages to traverse the network (upward and downward) increases by only two clock pulses (1.2 microseconds), due to network delay time. Because of the synchronous clocking used to push the messages across the network, pipelining of the message bytes results. Once the network delay interval has been overcome, message bytes arrive at their destination at the same rates regardless of how many components are in the configuration.

The second feature is the bidirectional nature of the network paths. As a message travels up the network tiers, it must contend with other messages. The result of this contention, taken globally, is that the highest priority message in the set of all messages launched simultaneously in the network will capture the apex node. Messages traveling down the network, from the apex node encounter no contention or collision from upwardly bound messages. It is then possible for the message to be broadcast to all processors on the network.

The third feature is the structure of network nodes. Each node has three bidirectional ports. Two ports receive data messages from (or broadcast data to) connected nodes on a lower tier of the network (or processors in the lowest tier). One port sends data to (or receives data from) its node connection on the next higher tier. Each node contains active logic which supports "test and send" functions. A node is therefore capable of determining which of two contending messages should be forwarded to the next higher tier. This decision process takes place independently of the configuration processors. If this were not so, a tremendous processing overhead burden would be placed on the AMPs and IFPs. Additionally, since the decision process is supported by dedicated, active logic using data received from high speed RAM circular buffers, performance is much faster than performing the decision operations in the processors.

The fourth feature of this design is the priority scheme implemented for the message traffic. When two messages simultaneously arrive at a network node (in the upward direction), the higher priority message is allowed to capture the node; i.e., pass through to the next tier. A "collision" signal is sent to the losing message's sender. The sender may attempt to re-transmit the message (compete again for network resources) once the winning message has completed its arrival at the apex node. This collision mechanism enforces protocol without use of an external control processor as discussed above.

This scheme is listed below in descending priority:

- First Arrival at the Network
- Lowest Command Code
- Lowest Key Field
- Shortest Key Field
- Lowest Data Field (Including Destination Status Field)
- Shortest Data Field

This priority scheme supports the message priority protocol listed in Table 4-1. These messages consist of two main groups: primary messages, originating from a requesting processor; and response messages. Primary

	COMMAND CODE	TAG	COMMAND DESCRIPTION
RESPONSES	00		NA
	01	OPID	NAK/LOCKED
	02	OPID	NAK/TN ERROR
	03	OPID	NAK/OVERRUN
	04	OPID	NA
	05	OPID	SACK/UNASSIGNED
	06	OPID	SACK/ASSIGNED
	07	OPID	SACK/BUSY
	08	OPID	SACK/WAITING
	09	OPID	SACK/SEND READY
	0A	OPID	SACK/RECEIVE READY
	0B	OPID	SACK/DONE
	0C	OPID	SACK/NON-PARTICIPANT
	0D	OPID	SACK/INITIAL
	0E	OPID	ACK
	0F	OPID	NAP
PRIMARY MESSAGES	10	TN0	STOP MERGE
	11	TN	DATA MESSAGE
	12	TN	DATA MESSAGE
	13	TN	DATA MESSAGE
	14	TN	DATA MESSAGE
	15	TN	DATA MESSAGE
	16	TN	DATA MESSAGE
	17	TN	DATA MESSAGE
	18	TN	DATA MESSAGE
	19	TN	DATA MESSAGE
	1A	TN	DATA MESSAGE
	1B	TN	DATA MESSAGE
	1C	TN	STATUS REQUEST
	1D	TN	RELINQUISH TN
	1E	TN	ASSIGN TN
	1F	TN	START MERGE

Table 3-1 Message Priority Protocol [Cont'd]

messages are tagged with an originating processor ID (OPID), while response messages are tagged with a global transaction number.

Taking into consideration the low-value-high-priority scheme, it is apparent that responses have a higher priority than requests. In the response category, it is also evident that NAK/--- messages, which indicate negative message acknowledgement, have the highest priority in the system. SACKs (Status ACKnowledgements) are next, followed by ACK (ACKnowledgement) and NAP (Non-Applicable Processor) respectively. In a parallel processing scheme, it is essential to ascertain the status of processing components before employing them in calculations, since malfunctioning units can corrupt the final results. Under the scheme presented in the patents, the fault status of one processor will receive priority over normal response messages of all other processors involved in the operation.

The priority scheme also allows the network to be effectively a global sort/merge device in which data arrives at a destination processor in an ordered sequence.

The network/processor interface design is another key feature. This interface contains high speed RAM which is divided into buffers (output message space and circular input and output buffers), a directory of data message pointers and responses, and selection maps containing hash, class selection, and Destination Processor ID (DPID) maps. As a message arrives at the Ynet interface buffer, its key fields can be compared against the selection maps to determine whether the message is applicable to the processor. If so, it is sent to the processor. If not, a Non- Applicable Processor message code is launched into the network by the interface. This scheme allows messages to be sent to all, one, or a group of processors with no intercession from a control processor. Only the content of the message is needed to determine the destination. Also this design allows the "transparent" sending of messages between specific processors. This type of activity is performed during global deadlock detection, where a lead AMP requests other AMPs to search for blocked transactions and receives results.

4.1.2 Patent Study Summary

There are other significant features in the patents which have not been included in this section for the sake of brevity. What can be said in general after studying the patents is that the design backbone of the DBC/1012 advances the state-of-the-art in multiprocessor integration. The components themselves (processors, communications devices, etc.) are off-the-shelf in the present DBC/1012, which minimizes technical risk, and additionally allows the incorporation of highly reliable hardware devices. The design is actually hardware independent, except for the elementary logic components. The philosophy underlying the design can be considered application independent, since it can be the basis for multi-systems designs for many different applications requiring high speed computation, high throughput rates, and fault isolation/recovery. The requirement for a dedicated controlling processor and its associated heavy overhead is avoided by the architecture of the network and its inherent priority scheme. Applications requiring such an arrangement of subsystems, such as SDI-related programs may benefit from this design philosophy.

4.2 Benchmark Performance Tests

Mc^2 did not conduct benchmark performance testing during the course of this effort, since the contract did not specify access to a DBC/1012 configuration. However, results from previous performance testing activities were collected. These formed the basis for performance assessment in this section. Other performance results, collected from Teradata customers, will be discussed in the next section of this report.

Teradata had two objectives in mind in performing benchmark testing. The first was to demonstrate the performance of the DBC/1012 as compared to typical host/DBMS systems. The second, and perhaps more important objective, was to demonstrate that linear performance improvements could

be realized by increasing the size of a DBC/1012 configuration. Linear performance improvement can be defined as follows:

> Processing performance improvements are directly proportional to the number of processors participating in the specified operation.

In other words, if the number of processors is doubled, applications would be expected to run twice as fast. This means that there is no degradation of parallelism due to context switching or other sources of overhead involved in introducing additional processors to the configuration. What this also implies is that the distribution of data across processors remains uniform; i.e., a redistribution of data must occur when the system is reconfigured.

Reconfiguration of data does indeed occur when additional processors are added to (or deleted from) the system. The Data Dictionary/Directory on the DBC/1012 is copied to all processors. When processors are added to the system, the DBC/1012 operator runs a Reconfiguration program which redistributes the hash buckets for each AMP and then performs the data redistribution across all AMPS according to the new configuration specifications. This is performed entirely under utility software control. More detail on reconfiguration is found in Section 6.3 of this report.

4.2.1 The Liberty Mutual Benchmark

The Liberty Mutual Benchmark consisted of a series of operations on five large insurance tables comprising the test data base. This was a comparative test which involved three different configurations; a 12-IFP/24-AMP DBC/1012 hosted by an IBM 4341, IBM's Database2 (DB2) DBMS hosted on a large IBM 3083, and IBM's Structured Query Language/Data System (SQL/DS) hosted on a smaller 3083. (IBM 3083 systems are considered small to medium mainframe systems, while the 4341 is a supermini class system).

The 3083s used in the test (according to the Teradata benchmark briefing) were 5.5 and 3.1 MIP rated machines. The DBC/1012 benchmark configuration was rated at 14.4 MIPs by Teradata. This figure was calculated by multiplying the thirty-six DBC/1012 processing components by 0.4 MIPs, the processor rating for the Intel 8086 microprocessor employed in the AMPs and IFPs.

Tests performed were as follows:

A. Table Creation - Each of the five tables

B. Table Loading

C. Secondary Index Creation on Table 0

D. Table Loading (included comparison with FOCUS DBMS)

E. Process 1095 Inquiries (Using Table 0 and JOIN Function)

F. Process Three Reports (Using JOIN and Various Tables)

Results are tabulated in Table 4-2. Overall, the DBC/1012 outperformed the other DBMS/host configurations. There was an instance where the DBC/1012 was slightly slower than the DB2/3083 contender. This was in the creation of Report Number 2 in Test F. However, for the other two reports, DB2 ran much slower than both the SQL/DS and the DBC/1012 contenders. The discrepancy may be attributable to an opportune format specification for Report Number 2 with respect to DB2's DML and report formatting capabilities.

Another area which this benchmark addresses is the relative cost per MIP of the contending configurations. The 3083 B16 on which DB2 ran is quoted in the benchmark material as costing $354,386/MIP, and the 3083 E16 has a

TEST \ DBMS/RESPONSE	DBC/1012	DB2	SQL/DS	FOCUS
A. (5 Tables)	2 min.	15 min.	40 min.	---
B. (5 Tables)	3.2 hr.	6.6 hr.	12.8 hr.	---
C. (Table 0)	2.7 min.	15.5 min.	25.5 min.	---
D. (5 Tables)	3.2 hr.	---	---	4.4 hrs.
E. (Table 0)	0.16 min.	0.92 min.	1.00 min.	---
F. (Various Tables)	1.5 min.	19.9 min.	5.1 min.	---

Table 4-2 Liberty Mutual Benchmark Performance Times

cost of $451,613/MIP. The DBC/1012 configuration (4-IFP/4-AMP) was quoted at $100,262/MIP (this cost did not include the host; on the other hand, it did not reflect the new lower prices currently in effect from Teradata).

4.2.2 The 60-Processor Benchmark

On March 4, 1985, Teradata concluded a demonstration of a 20-IFP/40-AMP DBC/1012, the largest configuration yet assembled. This 1.7 million dollar system was rated by Teradata at 24 MIPs and had 19 gigabytes of unformatted disk capacity (one DSU per AMP). The purpose of this demonstration was to prove that Teradata's proprietary processing technology could provide a linear growth of processing capacity. (Another purpose, more implied than stated, was to demonstrate that a very large number of processors could work effectively as an integrated unit on a processing problem).

Demonstration benchmarks were run against several DBC/1012 configurations ranging from 6 to 60 processors. As more were added, linear increases in throughput and performance were achieved.

The data bases which were used were furnished by Teradata clients. The largest data base contained over 21 million rows of data, approximately 4 gigabytes in total size. Other tables were 1 million rows, 5.5 million rows, and 0.5 million rows.

Two hundred thousand prime key transactions (65% retrievals and 35% updates) were performed against the 1 million row table on one test series. Transactions per second were plotted against number of processors in a DBC/1012 configuration, resulting in a linear graph. Processing rates ranged from 15 transactions per second (6 processors) to 139 for the 60-processor configuration.

In a row qualification test requiring full file scan (no indexes), which also showed linear performance improvement as the configuration was increased, the 60-processor system processed 8333 rows per second on a 5.5 million row table.

In some cases, performance was not observed to be linear due to discrepancies in the even distribution of table rows to DSUs as the configuration changes. Teradata points out that queries that show relatively few rows will not show exact linearity for the single user. However, multiuser operations do experience performance linearity as the system grows. Part of this benchmark demonstrates this effect:

# PROCESSORS	# JOBS	PROCESSING TIME
20	1	128 minutes
40	1	60 minutes
20	3	285 minutes
40	3	142 minutes

4.2.3 Analysis

The benchmark tests for which results were provided clearly show the performance capabilities of the DBC/1012 and its robust architecture. They also demonstrate that there is virtually no loss of paralelism as the system grows to accommodate increases in data base processing requirements. They are, however, publicly releasable and must be regarded as advertisements for the system (and well they should be). Conditions for performing the benchmark tests were controlled to a limited extent. For example, queries in the prime key test were constructed under BTEQ, which takes advantage of the request caching feature, thereby speeding up data base command parsing by at least a factor of 3:1. Additionally, the processing of tables consisting of millions of rows lends to the probability of a more even row distribution across the DSUs. On the other hand, that is exactly the intent of the machine: to efficiently and quickly process operations involving large data bases. Another factor is

in sizing the configuration used in the benchmark tests. In none of the benchmark tests was there a reference to an optimal system configuration. Also, testing of any configuration approaching the maximum 1024 processor system limit has not been possible (there may not be 1024 processors in existence, let alone housed in one facility). The modeling activities reported on in Section 5 are currently the most feasible means of examining maximum system behavior.

It should be noted that what these tests proved was that the DBC/1012 integration scheme does not impose increasing control overhead penalties as the system grows. In this sense, its performance is linear. The transaction types selected for the test were such that the data access algorithms were themselves linear. In Section 5, we discuss at length, the effects of other transaction types which, because of the implemented data access algorithms, do not exhibit linear performance improvements with increases in configuration size. The first of these is retrieval on secondary keys. In the current implementation such a request is broadcast to all AMPs, thus increasing the number of AMPs does not reduce the searching load of any single AMP. Teradata plans for the next release include a change in the algorithm which will send secondary key requests only to those AMPs which can satisfy them. This will be much the same as a primary key operation, and will be linear. Certain JOIN operations, it appears, become less efficient as the number of AMPs increases, due to AMP-to-AMP communication for rows not locally JOINed at the AMP. This gives a negative component to the performance curve, but is so small that its effect is minimal.

4.3 Customer Survey

The DBC/1012 has yet to gain wide acceptance in the data base processing marketplace, since it is a new product in the recently developed product area of data base machines. Customer reaction to the DBC/1012, accordingly, has not been as public as for the popular DBMSs. Few trade magazines encountered during this study have addressed the DBC/1012, much

less customer reactions. Therefore, direct contact with DBC/1012 customers was considered essential in assessing the machine's behavior in operational environments.

Teradata provided the project team with a list of DBC/1012 installations and points of contact. An additional customer was contacted during the course of this task. Attempts were made to reach all customers by phone. Eight of the eleven customer contacts were reached, and of these eight, three were willing to complete a customer survey form. A fourth customer provided considerable technical information in a series of phone conversations. Currently, two completed forms have been returned.

Overall, customer response to the DBC/1012 has been extremely favorable across all areas including machine performance, ease of installation, reliability, and quality of Teradata engineering support.

The next paragraphs detail specific responses received from those customers providing substantial information. These customers are identified as Customer A, Customer B, and Customer C.

4.3.1 Customer A

Customer A installed a 2-IFP/8-AMP configuration which is intended to support securities market surveillance applications.

The DBC/1012 supports an IBM 370/3033 system under OS/MVS. COBOL/ISPF will be used for interactive programs with TEQUEL supporting data loading and manipulation operations. Installation of the DBC/1012 was "very smooth." Since the application data base was not relational, it is being rebuilt to the DBC/1012's specifications.

No performance data was obtainable on the installed system, since the data bases are currently being built. However, the Teradata Benchmark Facility system (4-IFP/20-AMP) was used to collect performance samples on a 2.8 million record customer file (table). The keys for this table were as follows:

- Keys: Trade-Date, Class-Symbol, ACCode, Origin-Code, Series
- Primary Key: All of the above fields except Series
- Secondary Keys: Each of the above fields except Origin-Code

Several queries were performed on the table using conbinations of primary and/or single/multiple secondary keys. Primary key retrieval response times varied from four to twenty-five seconds. Additional search constraints sped up the response time by reducing the number of rows found. Secondary keys took from ten seconds (for very selective indices) to over twenty minutes (for a non-unique key resulting in over 100,000 hits). The benchmark results indicated that uniqueness (or near-uniqueness) of keys had a positive effect on performance.

Customer A is in Year One of a three-year DBC/1012 assessment project. To date, they are pleased with most aspects of the machine, including its performance, the relational data base architecture, its fault tolerance, and its ease of installation. Minor quirks in the software have been noted and have been fixed by Teradata customer support personnel. TEQUEL is regarded as a "mediocre" fourth generation language.

4.3.2 Customer B

Customer B installed an 8-IFP/16-AMP configuration which is intended for banking market analysis.

This configuration supports two IBM 3081K MVS/XA machines which also employ CICS/VS. Development languages include COBOL, PL/1, and BAL. Presently, COBOL and PL/1 Preprocessor-developed software will account for approximately half of the DBC/1012 user workload. It is also estimated that 800 CICS users will interface to the system, with 300 users operating concurrently.

Installation of the DBC/1012 hardware took six man-hours and host software installation required four man-hours of effort. Initial data base loading times were unavailable.

Performance times were also unavailable since the implementation is in its early stages of development. However the following performance figures are estimated, based on a prototype data base consisting of 12 million rows at 580 bytes each:

- o Prime Key Operations:
  - Queries: 64/sec.
  - Updates: 48/sec.
- o Secondary Key Operations:
  - Queries: 32/sec.
  - Updates: 24/sec.

The DBC/1012 has been highly fault tolerant: two failures (attributed to faulty IFP processor chips) were detected during a two month period. Successful restart and recovery operations were initiated automatically.

Security measures will be upgraded by this customer in 1986 which will [illegible] password [illegible] automatic expiration/modification of passwords. Security [illegible] are [illegible] superior to those provided by [illegible], [illegible], [illegible], [illegible], and IBM.

Hardware and software support are regarded as excellent. One host software bug was encountered but fixes were received on a normal periodic software maintenance tape. Customer B is favorably impressed by both the Field and System Engineer support staff.

Training courses were judged to be well organized and reference materials were rated as excellent, except for the CICS Interface Manual (which is being redeveloped).

In all, the DBC/1012 was assessed relative to other DBMSs as follows:

- o Superior:
  - Sharability (multiple CPUs)
  - Throughput
  - Flexibility of Data Structure
  - Performance (relative to other relational DBMSs)
  - Restart Characteristics
  - Security
  - Ease of DBA Support
- o Competitive:
  - Query Language Friendliness
- o Inferior:
  - Traditional Rollforward/Rollback Recovery Capabilities

4.3.3 Customer C

Customer C, located in the Data Systems Division of a large defense contractor, is performing a long term, extensive assessment of the DBC/1012 for potential implementation in operational systems development projects.

Their 8-IFP/12-AMP configuration has supported up to three mainframe IBM systems concurrently. Applications are oriented to COBOL programs running under the CICS environment.

Because this configuration does not support a specific application, emphasis has been placed on the study of DBC/1012 data base operations, instead of the normal data base reorganization/applications building activities performed by other customers. This approach, in essence, has furnished Teradata with a Beta-site type relationship with the customer.

Customer C has provided the Project Team with detailed information on various aspects of DBC/1012 operations which are discussed below.

4.3.3.1 CICS Operations

The DBC/1012 fully supports the closed-ended transaction and conversational methods of CICS operation. Closed-ended transactions allow the user to obtain single answers and then return control to the CICS control program. Conversational transactions allow menu-in, menu-out processing to continue under the control of a single applications program, with no exit to the CICS executive. Conversational transactions, however, consume a great deal of machine resources, such as main memory and protracted file locks. A pseudo-conversational transaction method is sometimes used in place of conversational mode to free resources between transactions. Under this mode, results from a prior program can be used as inputs to a subsequent program, since control blocks can be saved. Excessive computer resources are not held between program executions as they would be under conversational mode. The DBC/1012 does not not fully support pseudo-conversational mode. The Task Termination Exit, provided by Teradata will save control blocks and allow the specification of the subsequent program. However, for each program series (i.e., string of programs which are to process a data stream), separate sessions must be invoked by the CICS user. Multiple program series executions under one

session will not work, since the DBC/1012 will terminate the session and free all resources once a program series has been completed. Customer C has related these findings to Teradata.

4.3.3.2 Spooling

During retrieval operations, an AMP spools all multi-row results. All rows from the AMP's DSU(s) which satisfy the request must be retrieved and the spool file must be closed before the AMP can return its results. This gives the impression that DBC/1012 processing may be slower than some DBMSs since, in comparative benchmark tests, results from another DBMS will start to appear on the user display screen sooner than DBC/1012 results. Teradata is addressing this problem on two fronts. The first is that, in future releases, if data retrieved at an AMP is less than 32K words, no spool file will be created: data will be sent up the Ynet as it is found. The second, and more far-term solution may lie in a parallel spool file input/output processing arrangement.

4.3.3.3 Performance

Customer C has been favorably impressed with the performance of the DBC/1012. In some cases, it has been outperformed by DB2 running on a 3090 Sierra at the facility. Several reasons account for these results. First, the 8/12 DBC machine is rated at only 5 MIPs while the Sierra is rated at 30 Mips. Secondly, the DB2 can precompile requests while the DBC/1012 must parse requests at the IFPs. If TEQUEL macros are used in the tests, parsing is minimized and data can be passed to cached requests, thus reducing the performance gap. During large retrievals, however, the DBC/1012 actually outperformed the DB2/Sierra. Third, the IBM can access data from its disks over a 3 MB/second data channel while the IFPs currently support 500 KB transfers.

Data base loading performance was also satisfactory. Data bases on the DBC/1012 could be loaded as quickly as loading an IBM 3380 disk subsystem under DB2.

4.3.3.4 Overall Comments

Customer C is very pleased with the DBC/1012 for the following reasons:

- It arrived in working condition with fully operational hardware and software.
- Price/performance: Comparable (the 8-IFP/12-AMP configuration) or superior performance to DBMS/host configurations costing far more.
- The responsiveness of Teradata in providing updates, discussing fixes and potential approaches, and their general attitude in striving to improve all aspects of the DBC/1012.

4.4 Hands-On Experience

During the 8 - 12 April data collection trip to the Teradata Corporate Headquarters, a brief on-line session was arranged on the DBC/1012 to acquaint Mc^2 with first-hand operation experience. The brevity of the session (approximately one to two hours) was not due to system operating schedules, but rather to the extensive periods of time spent on discussing machine architecture, DBMS operation, data base structure, and fault tolerance issues. On the average, eight to ten hours per day were spent with Teradata personnel discussing all aspects of the system.

Several types of data base requests were exercised on the system, ranging from simple (prime key) retrieval/updates to complex joins on a demonstration marketing data base consisting of two tables: CLIENTS and SITES. Operations were not precisely timed, since no knowledge of host processor activities or other users was available. However, times were in line with those listed in the customer survey forms for similar operations.

A highlight of the on-line session was the EXPLAIN facility. The EXPLAIN facility outputs text to the terminal which describes what the TEQUEL optimizer will do in the performance of a data base request. EXPLAIN lists the steps which compose the operation and predicts the number of records which will be targeted by the request. EXPLAIN allows the user to know in advance the effects of a request and can aid in preparing efficient data base requests. Under an ITEQ session, EXPLAIN has the following format:

EXPLAIN {TEQUEL request}

EXPLAIN is especially useful in assessing the effects of a JOIN operation. During the on-line session, EXPLAIN provided information on the left and right JOIN table components, stated the join step form employed (JOIN forms are: Product, Exclusion Product, Merge, and Exclusion Merge), and predicted the number of hits on which the JOIN would be made. EXPLAIN is also used by the DBC/1012 DBMS developers to examine enhancements to DBMS operations. No documentation was found describing EXPLAIN in the manuals acquired by Mc^2.

In summary, the brief ITEQ session was user friendly and demonstrated the ease with which data base requests could be generated, especially by a user with a general working knowledge of TEQUEL.

4.5 Reliability

This section details reliability statistics for the DBC/1012. An informal Failure Mode and Effect Analysis (FMEA) was also performed to indicate possible modes of failure of each DBC/1012 component and its effects on overall system operation.

4.5.1 Reliability Statistics

Two internal memos generated by Teradata and made available to the Mc^2 Project Team provided the following reliability statistics.

4.5.1.1 Memo #1

A six month study of a sample configuration which included 24 disks and 118 boards yielded the following results:

	FAILURES	INSTALLED OPERATING HOURS	IMPLIED MTBF
DISKS	3	100,000	30,000
BOARDS	6	500,000	80,000

4.5.1.2 Memo #2

The following table indicates hardware (HDW) and software (SFT) errors recorded at five customer installations:

	SEVERITY 1 FAILURES	
CUSTOMER	HDW	SFT	CONFIGURATION
A	4	1	2x4
B	2	2	4x8
C	5	2	8x16
D	2	0	2x4
E	2	0	4x8

For purposes of this analysis Severity One hardware failures are those for which a Customer Engineer (CE) was needed to effect a repair. None of the hardware failures listed above were system-wide in nature. All of the failures were local to a particular processor. Given that the "Fallback" option is elected, a customer would experience outage totaling the amount of time for a system restart, (8 minutes). Severity One software failures are those that rendered the system unusable by the customer pending a fix or work-around. To date Only one Severity One disk failure has been experienced in the field. Here again, this is still a processor local failure and in this instance the system was up while repairs were effected on the down disk unit. Current hardware MTBF for a processor module is 23,800 hours. A processor module consists of the following:

- Memory Printed Wiring Assembly Board (PWBA)
- Processor PWBA (AMP or IFP)
- Two Ynet PWBAs

4.5.2 Failure Mode and Effect Analysis

The analysis is presented on a component-by-component basis with emphasis placed on the IFP, AMP, DSU, and Ynet components of the DCB/1012 configuration.

"Criticality of Failure", is a topic included in this analysis. For the purpose of this report, it is assigned stated values according to the following rules:

- Minor: Users of DBC/1012 area (selected tables or data bases) experience slight degradation in system performance.

- Significant: All DBC/1012 users experience slight degradation in system performance.

- o Major: Users of a DBC/1012 area cannot continue processing; Areas are unaccessible to their applications.

- o Critical: All DBC/1012 users encounter unaccessible areas. Normal activity on a system-wide basis is impossible.

4.5.2.1 IFP FMEA

4.5.2.1.1 Description of Failure Modes

Hardware failures or Parser failures.

4.5.2.1.2 Anticipated Cause of Failure

Hardware failure in IFP processor board components (e.g., memory errors, interface failure, etc.).

4.5.2.1.3 Possible Effects of Failure

All requests to and responses from the data base attempting to use the failed IFP are blocked. If the IFP goes down, the system will restart. Sessions on the down IFP will be reassigned to active IFPs. In progress work will be lost.

4.5.2.1.4 Criticality of Failure

Significant if IFP fails. Degraded mode of operation due to increased load on remaining IFPs. Critical if all IFPs fail. All data base operations on the DBC/1012 configuration are halted.

4.5.2.1.5 Possible Corrective Action or Preventive Maintenance

Repair or replace the IFP. Reconfiguration is handled automatically.

4.5.2.2 AMP FMEA

4.5.2.2.1 Description of Failure Modes

Soft errors and hard errors.

4.5.2.2.2 Anticipated Cause of Failure

Failures in the processor board components of the AMP module.

4.5.2.2.3 Possible Effects of Failure

Inaccessibility of rows in non-fallback data bases tables. If an AMP fails and the Commit Point has not been reached, the existing data is protected. When the AMP restarts, it refreshes the Transient Journal in memory and backs out all aborted transactions.

4.5.2.2.4 Criticality of Failure

Minor to major, depending upon the following circumstances:

- If no fallback had been specified for the table, its elements residing on the down AMP are inaccessible. Once the AMP is brought back on line, the data can be recovered (Minor/Major failure depending on whether data was committed from Transient Journal to the data base).

- If fallback had been specified, data base information is recoverable from the fallback AMP. A minor failure. The degree of degradation is dependent upon the number of remaining AMPs which carry the fallback data. Degradation can be computed as:

  $$D = 1/n$$

  where n is the number of AMPS in the cluster.

- If two or more AMPs assigned to the same cluster fail, table data is inaccessible to operational users (same effect as a single no-fallback AMP failure).

- The failure of two or more AMPs assigned to separate clusters is considered as a series of minor errors, resulting in degradation of performance for users on each affected cluster.

4.5.2.2.5 Possible Corrective Action or Preventive Maintenance

Repair the AMP module, or use the reconfiguration program to assign replacement AMPs. Use the fallback option during data definition operations to ensure that data is not lost during a single AMP failure. Use the clustering facility during configuration to decrease the probability of primary and fallback AMP failure.

4.5.2.3 DSU FMEA

Since a DSU and an AMP are considered a single logical unit under a DBC/1012 configuration, FMEA is virtually the same as it is for the AMP. Differences exist only in the descriptions and causes of failure. The main difference is that DSU failure will compromise data where an AMP failure won't. Data must be recreated. A major failure occurs on non-fallback information, while failure on fallback data is considered minor.

4.5.2.4 Ynet FMEA

The Ynet is the most vulnerable point in the configuration, since the number of Ynet circuits in a DBC/1012 is fixed at two.

4.5.2.4.1 Description of Failure Modes

Hardware failure.

4.5.2.4.2 Anticipated Cause of Failure

Clock, circuit level, etc.

4.5.2.4.3 Possible Effects of Failure

AMP-to-AMP and AMP-to-IFP communication is impossible on failed nodes of the Ynet circuit. Broadcasting and merging operations will be performed on the remaining Ynet. A Ynet will not operate if it is partially down.

4.5.2.4.4 Criticality of Failure

Two redundant Ynet circuits are provided in a DBC/1012 configuration. If one Ynet circuit fails, the other will automatically handle the load under a degraded mode of operation. Thus the failure is regarded as significant.

If both Ynet circuits fail, system-wide activities are halted. A critical failure situation exists.

4.5.2.4.5 Possible Corrective Action or Preventive Maintenance

Replacement of Ynet circuitry.

4.5.3 Power Outages

To prevent possible single points of failure, recommendations are given in the Installation Manual on connecting DBC/1012 components to power sources.

4.5.4 Reliability Assessment Summary

We consider the DBC/1012 to be a highly reliable system based on customer survey responses and our Failure Modes Effect Analysis. The redundancy of both hardware and software components throughout a configuration minimizes the effects of failure and permits non-stop running under most failure events.

We also consider the DBC/1012 a highly maintainable system, since it is fabricated with off-the-shelf, universally-available components, such as the Intel processor and FUJITSU/CDC disk drives. Customers have indicated that field engineering support has been both timely and effective.

SECTION 5
PERFORMANCE MODELING

This section describes an analytic model for analyzing DBC/1012 performance under various workload conditions using a range (minimum to maximum) of possible configurations. The layout, assumptions, calculations, parameters and a discussion of the results of the automated DBC/1012 analytic model are presented. The model was implemented on the Columbia 1600 PC desktop computer, using the LOTUS 123 spreadsheet package. It can be run on the Columbia portable PC and the IBM PC.

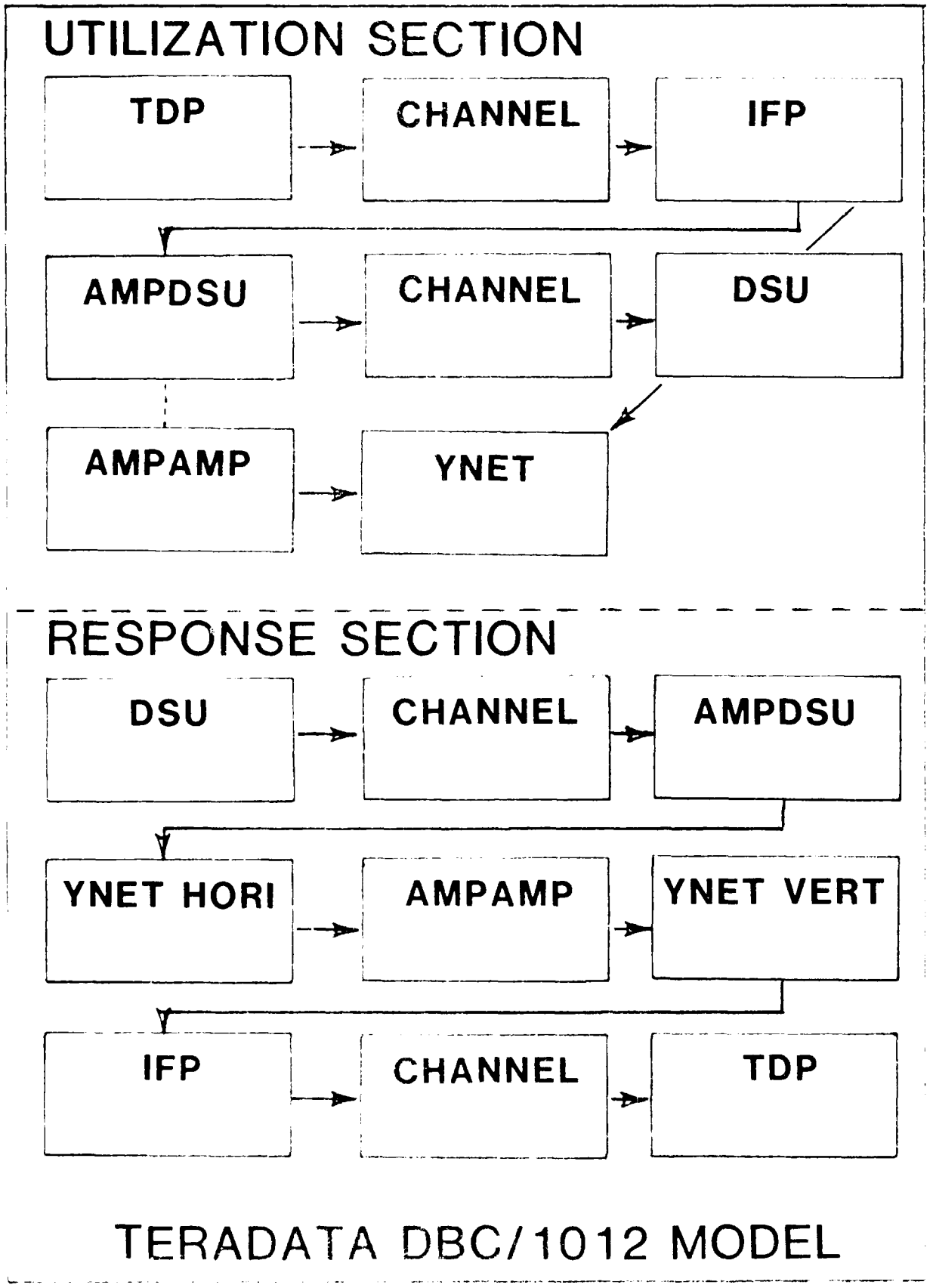
5.1 Mc2 Analytic Performance Model Description

The model, summarized in Figure 5-1 and illustrated in Figure 5-2, is composed of three general sections:

- The Utilization section, made up of the 8 component boxes in the upper half of the page;
- The Response section, made up of the 9 component boxes in the lower half of the page, and;
- The Parameter section, made up of the 19 labeled parameters at the bottom and in the lower right quadrant of the Utilization section, plus four labeled parameters in the upper left quadrant.

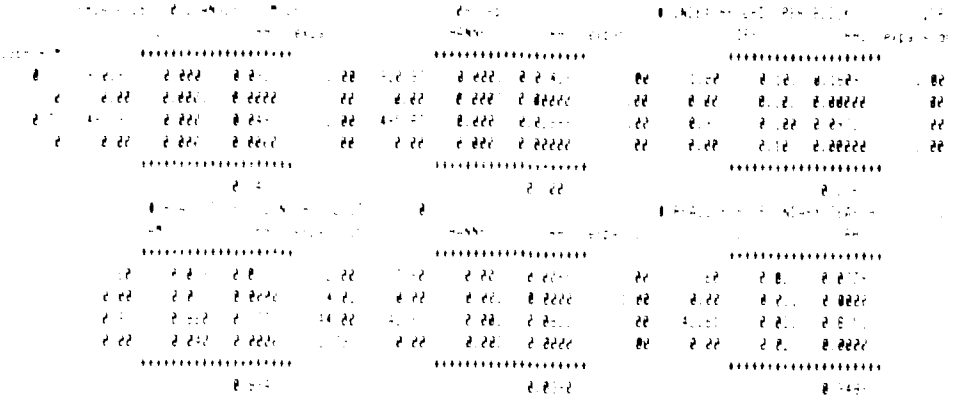
5.1.1 Layout of Component Boxes

The component boxes are constructed roughly in a HIPO format - input, process, output, and are arranged in a hierarchical sequence to facilitate

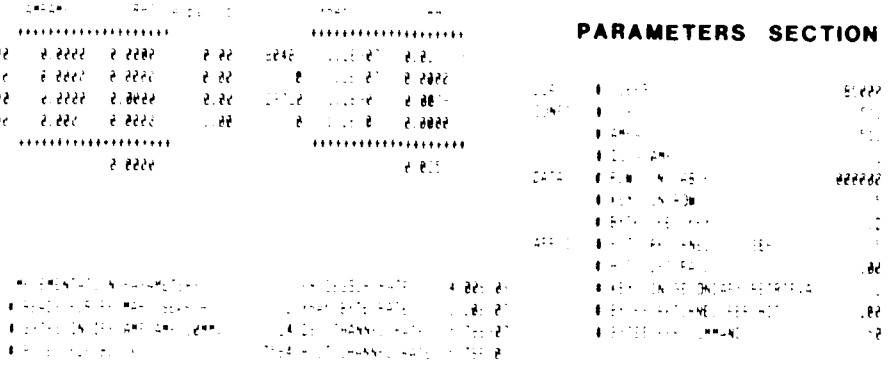


TERADATA DBC/1012 MODEL

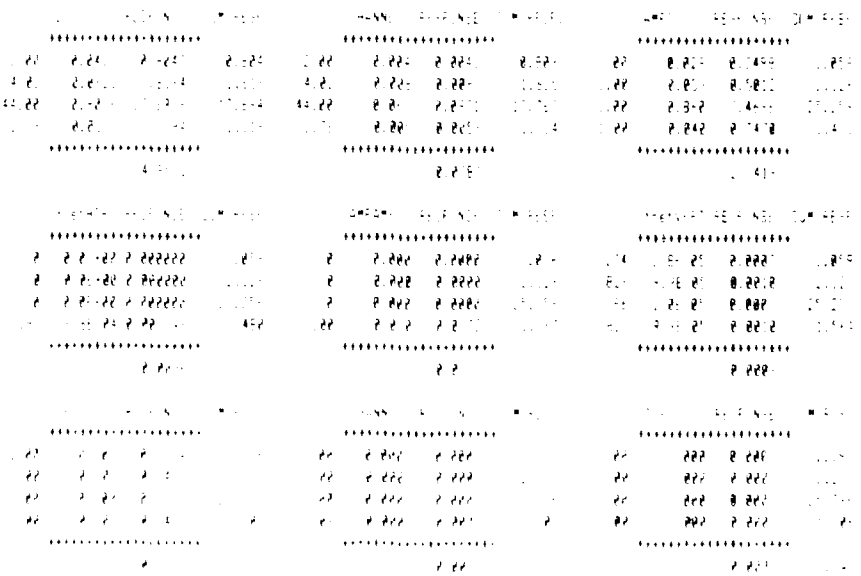
UTILIZATION SECTION



PARAMETERS SECTION



RESPONSE SECTION



progressive decomposition on the functions. There are at least three columns to each box, most have four (reference Figure 5-2):

- Column 1 - LAMBDA. Arrival Rate. In the Utilization section, this column is the number of requests per second arriving at the device. In the Response section, it is the number of units of work that the particular device must perform to satisfy one user request.

- Column 2 - Ts. The time to service one unit of work.

- Column 3 - RHO or RESPONSE. In the Utilization section, the calculated utilization of the component based on the arrival rate and the service time. In the Response section, the calculated response time for the device to perform the work associated with one user request.

- Column 4 - In the Utilization section, this is the "Expansion" column - the ratio of derived requests to entering requests that are sent to the next component. In the Response section, this column holds the cumulative response time as a request moves through the system.

The Expansion column is the vehicle for performing decomposition of the requests. For example, the value "2.00" in the Expansion column of the "AMPDSU" box means that for every request received by the AMP, two are generated for the DSU. A more detailed discussion of these columns and the source of their values is presented in Section 5.2.6.

There are five rows in each component box. The first four rows represent four types of requests. In the current implementation, the rows represent: SEARCH ON PRIMARY KEY, SEARCH ON SECONDARY KEY(s), UPDATE ON PRIMARY KEY, and JOIN ON PRIMARY KEY AFTER SELECTION ON SECONDARY KEY(s).

The fifth row contains either a summation of the utilization for the job mix, in the Utilization section, or an average response time, in the Response section.

5.1.2 Utilization Section

The component boxes of the Utilization section are arranged in a top-down sequence to facilitate the decomposition of requests. There is a fork in this arrangement at the AMP (as indicated in Figure 5-1). Two AMP boxes have been created in order to support the separate assignment of resulting DSU request and AMP-to-AMP communication expansion values. Also, the Ynet box appears at the end of the list to facilitate the aggregation of "requests" to the Ynet imposed by both IFP-to-AMP and AMP-to-AMP activity.

The Parameter section shares the upper half of the form with the Utilization section. It will be discussed below. Three other labeled values appear in the Utilization section - "# INDEX RECORDS PER BLOCK". "# READS FOR SECONDARY SEARCH", and "Ynet Levels". These values are derived from other parameters and are placed where they are to comply with the calculation order constraints imposed by "123".

5.1.3 Response Section

The component boxes of the Response section are arranged in a bottom-up sequence to facilitate the accumulation of response times. An additional Ynet box (reference Figures 5-1 and 5-2) has been introduced to simulate the order of processing in which AMPs communicate horizontally with each other (YnetHORI), before communicating vertically with the IFPs (YnetVERT). Response times for each type of request are calculated at each component box and are accumulated in the fourth column.

5.1.4 Parameter Section

The Parameter section, for user entry of parameters, is located in the lower part of the upper half of the form. An additional set of user supplied parameters appears in the top left portion of the form. The individual parameters are discussed below. Appearing in brackets after each parameter name is an abbreviation which will be used later in formula descriptions. Parameters are summarized in Figure 5-3.

5.1.4.1 User Parameters

The model employs the PROTECT feature of "123" to prohibit user entry of values into the model itself. Selected cells in the Parameter section have been UNPROTECTed to allow entry of user supplied parameters. Those cells appear on the screen as highlighted values, brighter than the PROTECTED cells. Each parameter is described below:

5.1.4.2 Load Parameters

These parameters are used to define the load in terms of the requests entering the system from users or from software.

USER RPM [RPM] - The four cells at the top left of the form are available for entry of the arrival rates of the four types of requests. These parameters define the mix of activity. Each should be expressed in terms of requests per minute per user. The request types are, from top to bottom:

- RETRIEVE ON PRIMARY KEY
- ANDed RETRIEVE ON SECONDARY KEY(s)
- UPDATE RECORD BASED ON PRIMARY KEY QUALIFICATION
- JOIN ON PRIMARY KEY AFTER SELECTION ON SECONDARY KEY

USERS [USERS] - The number of users submitting requests.

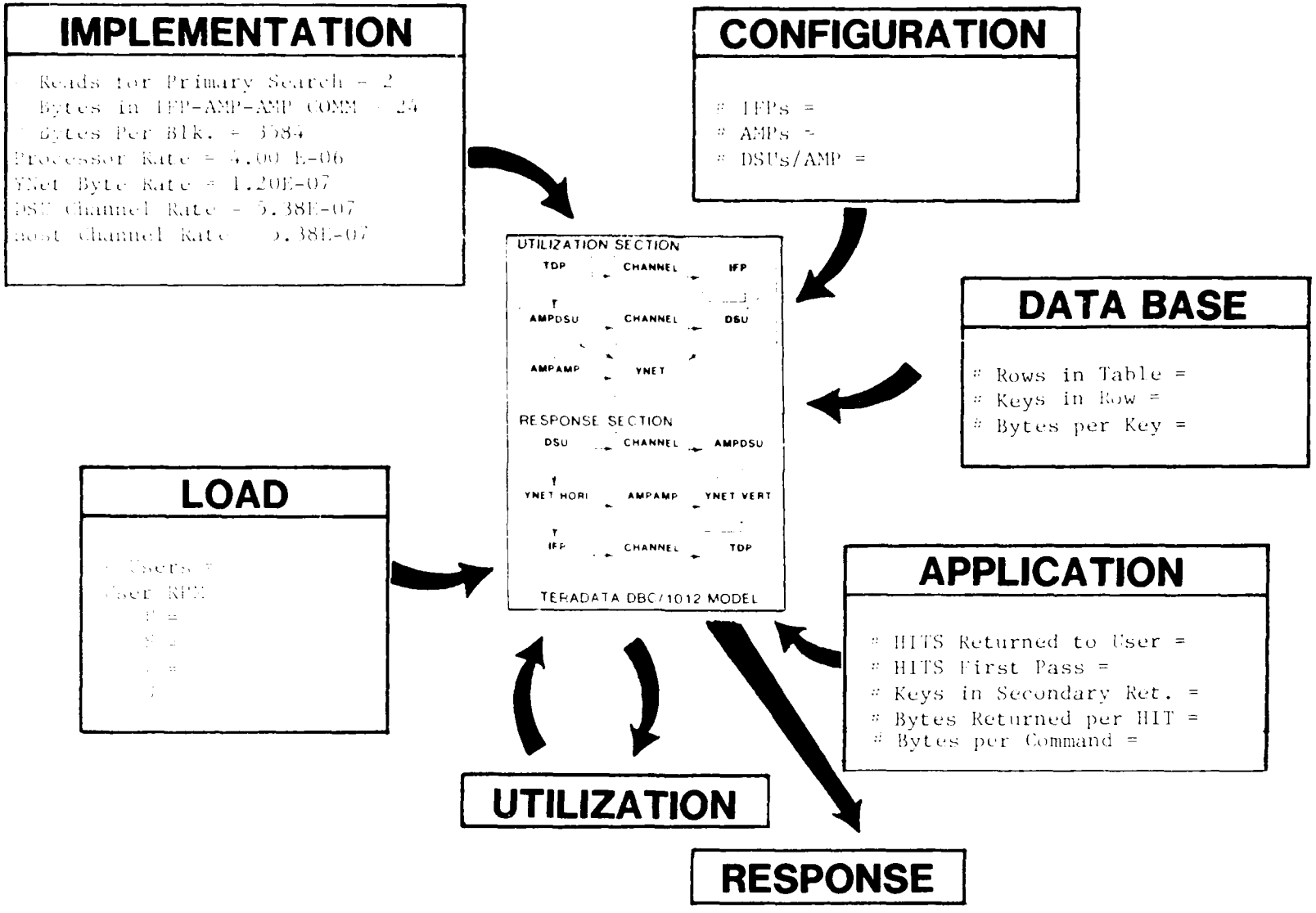


Figure 5-3 Parameters

5.1.4.3 Configuration Parameters

These parameters are used to define the configuration of the DBC/1012 being modeled.

IFPs [IFPS] - The total number of IFPs in configuration.

AMPs [AMPS] - The total number of AMPs in configuration.

DSUs/AMP [DSUS] - The number of DUSs attached to each AMP.

5.1.4.4 Data Base Parameters

These parameters are used to specify certain characteristics of the data base which are of importance to the model.

ROWS IN TABLE [ROWS] - The number of rows in the table which is assumed to be the one upon which secondary searches are performed.

KEYS IN ROW [KIR] - The number of indexed fields in the row upon which secondary searches are performed.

5.1.4.5 Application Parameters

These parameters reflect the general format of requests and their results. They are defined to be typical of the application being modeled.

HITS RETURNED TO USER [HITS] - The average number of qualifying rows returned to a user as the result of a secondary key retrieval.

HITS 1st PASS [H1ST] - The average number of hits per secondary key before intersection of hit lists.

KEYS SEARCHED [KEYS] - The number of secondary keys specified in an average user request.

BYTES RETURNED PER HIT [BRPH] - The average number of bytes returned per hit as the result of a user query request.

BYTES PER COMMAND [BPC] - The length of the average user request.

5.1.4.6 Implementation Parameters

These parameters represent certain assumptions concerning DBC/1012 system implementation.

READS FOR PRIMARY SEARCH [RPS] - The number of reads of the DSU required to retrieve a row on a primary key.

BYTES IN IFP-AMP-AMP COMM. [BFC] - The number of bytes in the average communication packet sent between IFPs and AMPs and between AMPs.

BYTES PER BLOCK [BPB] - The number of bytes in a data or index block on the DSU. This parameter is used to calculate the volume of data transferred from the DSU to the AMP for each read during a search, and the number of reads of the index subfiles performed during a search.

PROCESSOR RATE [PRATE] - The MIP rate of the AMPs and IFPs.

Ynet BYTE RATE [YRATE] - The time required for a byte to travel between two nodes in the Ynet.

DSU CHANNEL RATE [DRATE] - The time required for a byte to travel between two nodes in the Ynet.

DSU CHANNEL RATE [DRATE] - The time required for a byte to travel across the AMP to DSU channel.

HOST CHANNEL RATE [HRATE] - The time required for a byte to travel across the host to IFP channel.

5.1.4.7 Derived Parameters

These parameters are not normally changed by the user of the model. They are entered as formulas rather than as values, and represent our understanding of the storage and search algorithms employed by the DBC/1012.

INDEX RECORDS PER BLOCK [IRPB] - This value is used to calculate the rate at which the index sub-file is sub-divided during a B-Tree search. It is the number of index records in one data block. The formula is:

BPB/(BPK + overhead)

Overhead is 16 bytes: 12 bytes for the row header,
4 bytes for the field header.

READS FOR SECONDARY SEARCH [RSS] - This value represents the number of reads issued to the DSU when performing a B-Tree search on a secondary key. The basic formula is:

LOGm N where:
m = IRPB
N = RCWS/(AMPS*DSUS)

Since there is no LOGx function in "123", the following equivalent formula, using natural logarithms, was implemented in the model:

Ln N/Ln m, rounded up to the nearest integer.

Ynet LEVELS [YNL] - The number of levels in the Ynet tree required to support the configuration. The basic formula is:

LOG2 m where:

m = IFPS + AMPS

It is implemented in the model using natural logs as described above.

READS TO GET LONG HIT LIST [RLHL] - The number of reads to read in all secondary index sub-file records which meet the criteria during the first pass of a secondary search or JOIN.

(HIST/IRPB)-1

5.2 Functional Decomposition and Loading Calculations

This section presents the formulas, assumptions and procedures followed in the construction of the central model.

5.2.1 LAMBDA Values

The arrival rates (LAMBDA Values) appear down the lefthand side of each component box and are analogous to the "Input" of a HIPO chart. With a few exceptions, the LAMBDA of a component box in the Utilization section is the product of the LAMBDA and the "Expansion" factor of the preceding component box. The LAMBDAs in the Response section are individually calculated to represent the number of work units of a particular device required to satisfy one user request. The exceptions are as follows:

- TDP LAMBDA [TDPL] is the result of converting from single user requests per minute to total requests arriving per second. The formula is:

  RPM times USERS divided by 60

- AMPAMP LAMBDA [AMPL] is implemented only for JOIN functions (row 4 of the AMPAMP component box). Rows 1 through 3 are 0. This is based on the assumption for model purposes that the only significant AMP-to-AMP communication is that involved in JOINs. There is no logical antecedent component box for AMPAMP. The expansion of requests for AMP to AMP communication is determined in the AMPs themselves, therefore, its calculation has been included in the AMPAMP LAMBDA formula. The expansion component of Row 4 is the number of requests sent by an AMP to other AMPs, times the number of AMPs. This value is calculated as follows:

  - The number of hits at a local AMP:
    # H1ST divided by AMPS, minus

  - The number of hits satisfied by the local AMP:
    # H1ST at local AMP divided by AMPS

  - The above difference times number of AMPs.

  This is simplified to:

  H1ST - (H1ST/AMPs)

  and multiplied by TDPL.

- Ynet LAMBDA [YNLAM] is made up of all IFP-AMP and AMP-AMP traffic. The basic formula for calculating the traffic in bytes across the Ynet for each "message" is:

  YNL times 2, plus # bytes in message, plus fixed overhead of 16 bytes (2 byte command word, a key field (optional with data message), and a 2 byte destination select word [NEC-2: Sheet 8]).

If the established procedure for this model were to be followed, the Ynet LAMBDA would represent the number of messages arriving at the Ynet per second and the above formula would appear in the Ts calculations to give the time to process one message. There are, however, three types of messages arriving at the Ynet at differing rates, each of a different length:

IFP to AMP communication packets, represented by: BFC

AMP to AMP communication packets, represented by: BFC plus BPK

Row data being returned to the IFP, represented by: BRPH

For this reason, it was necessary to perform the calculation earlier, in effect expressing LAMBDA in terms of bytes, rather than messages. The formulas for each row, and their derivations, are as follows:

Row 1:

IFP-AMP communication:

TDPL (YNL*2 + BFC + 16), plus

AMP to IFP transmission of row data:

TDPL (YNL*2 + BRPH + 16)

Simplified:

TDPL(YNL*4+BFC+BRPH+32)

Row 2:

IFP-AMP communication:

TDPL (YNL*2 + BFC + 16), plus

AMP to IFP transmission of row data:

TDPL*HITS (YNL*2 + BRPH + 16 + [see note on page 5-12]

Simplified:

TDPL(YNL*2+BFC+16+(HITS(YNL*2+BRPH+16+[see note on next page]))

Row 3:

IFP-AMP communication:

TDPL (YNL*2+BFC+16)

Row 4:

IFP-AMP communication:

TDPL (YNL*2+BFC+16), plus

AMP-AMP communication:

AMPLAM (YNL*2+BFC+BPK+16), plus

AMP-IFP transmission of row data:

TDPL*HITS (YNL*2+BRPH+16+[see note on next page]

Simplified:

(TDPL+AMPLAM)(YNL*2+BFC+16)+TDPL*HITS(YNL*2+BRPH+16+[see note on next page]

o YnetHORI LAMBDA [YNHL] represents only the IFP-AMP traffic portion of Ynet activity required to satisfy one request. The formula is, for Row 4:

AMPLAM(YNL*2+BFC+BPK+16)

o YnetVERT LAMBDA represents only the IFP-AMP traffic portion of Ynet activity required to satisfy one request. In essence, these formulas are the same as those for Ynet LAMBDA, with the AMPAMP communication and the TDP LAMBDA factor removed. The formulas are:

Row 1:

YNL*2+BFC+BRPH+16+[see note]

Row 2:

YNL*2+BFC+16+HITS(YNL*2+BRPH+16+[see note]

Row 3:

YNL*2+BFC+16

Row 4:

Same as Row 2.

[Note: Add BPK if ORDERED option is being modeled].

All other Response section LAMBDAs are "1", except the first three rows of YnetHCRI and AMPAMP, which are "0", and DSU and the DSU channel, which are equal to the AMPDSU expansion value.

5.2.2 Ts Values

The service times (Ts values) are expressed as the time in seconds to perform one request. In some cases, this includes both the time to process commands and the time to move data block responses. In some cases, they are assigned as values; in others, they are calculated.

Utilization section:

- TDP Ts - currently assigned a value of .001

- TDP-IFP CHANNEL Ts

  Row 1:

  (BRPH+BPC)HRATE

Row 2:

(BPFH*HFTS+LPC)HRATE

Row 3:

(BPC*2)HRATE

[Note: The multiplication by two means that the update information embedded in the command will double the length of the command.]

Row 4:

Same as Row 2

o IFP Ts

Same as those above, except that PRATE is used in place of HRATE and a parsing rate is added to each.

Parsing rates must be changed in the formula depending on the type of activity being modeled - batch or interactive. They are as follows:

	BATCH	INTERACTIVE
Row 1	.1	.333
Row 2	.1	.667
Row 3	.1	.400
Row 4	.1	.667

o AMPDSU Ts

Row 1:

(IFB*BPC+LPC)PRATE

This represents processing the cylinder block, plus the data block, plus the command itself.

Row 2:

(BPB(RSS*KEYS+RLHL+RPS*HITS/AMPS)+BFC)PRATE

[Note: The secondary search calculation (RSS, includes only the number of reads to get the row identifier. After that, a primary search must be performed.]

Row 3:

(BPB(RPS+4*(RSS+2)*KIR)+BFC)PRATE

This formula is distilled from the following scenario based on the assumption that one row, specified by a unique primary key, is being modified, and that, as part of the modification, all of the secondary keys are being modified:

- Primary search to find and read the row to be modified
  # READS FOR PRIMARY SEARCH

- Write the modified row. One write.

- Find the secondary index entries to be modified
  # READS FOR SECONDARY SEARCH * # KEYS IN ROW

- At each node of the B-Tree, insert new pointer
  # READS FOR SECONDARY SEARCH * # KEYS IN ROW

  [Note: Write is equivalent to Read]

- At each node of B-Tree, delete old pointer
  # READS FOR SECONDARY SEARCH * # KEYS IN ROW

- Re-write the cylinder index. One write.

Row 4:

(BPB(RSS+RLHL+2(RPS)(HIST/AMPS))+BFC)PRATE

This formula is distilled from the following scenario:

- PERFORM a secondary search looking for local hits
  # READS FOR SECONDARY SEARCH

- Read the local hits
  # READS FOR PRIMARY SEARCH times the number of hits at the local AMP, which is H1ST/AMPs

- Extract primary keys and hash

- Read local hits
  # READS FOR PRIMARY SEARCH times the number of hits at the local AMP on the second table, which is H1ST/AMPS/AMPS

- Transmit primary key misses (for AMPAMP formula, later)
  H1ST/AMPS - H1ST/AMPS/AMPS

- Receive primary key misses from other AMPs (for AMPAMP formula, later)
  (AMPS-1)(H1ST/AMPS-H1ST/AMPS/AMPS)

- Hash on received misses from other AMPs

- Retrieve local hits
  (RPS(H1ST-H1ST/AMPS)AMPS

o AMP-DSU CHANNEL Ts

  BPB*DRATE

- DSU Ts

  Access time - .018

  plus,

  Delay for transfer:

  BPB/(# bytes per track (23442) * 60)

[Note: This assumes that the .018 given as access time in the literature includes head movement and latency.]

- AMPAMP Ts

  Zero for Rows 1 through 3

Row 4:

BFC*PRATE

- Ynet Ts

  Since the Ynet initiates no traffic of its own, and since the traffic has, by this stage in the model, all been translated into byte traffic, the Ynet Ts is equal to the Ynet byte processing rate - YRATE.

  Response section Ts values are equal to their Utilization section counterparts times the local LAMBDAs.

5.2.3 RHO Values

All RHO values are calculated according to the following formula:

LAMBDA * Ts

The RHO in Row 5 is the total RFC for the component. It is a simple sum of the Row 1, 2, 3 and 4 RHOs.

5.2.4 Response Values

All response values are calculated according to the following formula:

Ts/(1-RHO)

The RHO value used in the calculation for a given component box is the total RHO (Row 5) for the corresponding component box in the Utilization section, with the following exemptions:

- The RHO used in both the AMPDSU and AMPAMP response calculations is the total RHO for the AMP, that is, the sum of the total RHOs (Row 5) for AMPDSU and AMPAMP.

- The RHO used in both the Ynet HORI and YnetVERT response calculations comes from the Ynet component box.

5.2.5 Expansion Values

The expansion values in the Utilization section indicate the number of requests a component box sends to the next component box in sequence. They are analogous to the output of a HIPO chart, with the difference that the expansion represents the number of requests being sent to only one of the succeeding devices, rather than the total number of requests being issued. Thus, the expansion from the channel between the TDP and the IFP component boxes is 1 divided by the number of IFPs; the Row 1 expansion from the AMP to the DSU represents the number of read requests generated in response to a search on primary key divided by the number of DSUs per AMP. Following are descriptions of all of the expansion values.

- TDP to CHANNEL expansion

  For all Rows, the expansion is 1. The TDP is merely passing on requests, one for one.

o CHANNEL TO IFP expansion

For all Rows, the expansion is 1 divided by the number of IFPs. When the TDP passes the requests to the Channel, it addresses each to a particular IFP.

o IFP to AMPDSU expansion

The expansion is the ratio of IFPs to AMPs. It is assumed that an IFP, when performing an operation involving a primary key, sends the request to the AMP which is responsible for the particular key value appearing in the request, and will therefore send the request to only one of the AMPs. This would normally be expressed as 1 divided by the number of AMPs (in other words, a primary key operation goes to only one AMP), but since the general procedure for deriving LAMBDA is to multiply the expansion value by the sending device LAMBDA, and the LAMBDA for the IFP is the result of dividing 1 by the number of IFPs, that factor must be restored.

o AMPDSU to CHANNEL expansion

These expansion values represent the number of Read/Write requests generated by the AMP for the DSU(s). Each Row is different.

Row 1:

RPS

Row 2:

RSS*KEYS+RLHL+RPS*HITS/AMPS

For Row 3, the expansion is based on the scenario described under "AMPDSU Ts", Row 3. The formula is:

RPS+4*RSS*KIR+2

For Row 4, the expansion includes all of the activity generated by an AMP to perform its own secondary search and the primary searches associated with attempting joins locally, plus all of the primary searches levied on it by the other AMPs looking for JOINs to their hits. The formula is based on the scenario described under "AMPDSU Ts", Row 4:

RSS+RLHL+2*RPS*(HIST/AMPS)

- o CHANNEL to DSU expansion

  For all Rows, the expansion is 1 divided by the number of DSUs per AMP.

- o AMPAMP to Ynet expansion

  AMP to AMP communication is assumed to be significant only for JOINs. The expansion values for ROWS 1, 2, and 3, than are 0. The expansion value for Row 4 is "1".

- o There is no expansion for the Ynet.

- o The DSU has no expansion because it is at the end of the line.

5.3 Performance Testing Activities

Three different basic types of runs were made:

- o General DBMS Operations
- o C^3I Tests
- o Teradata Benchmark Tests

For each of these three basic types of runs there were multiple subtypes, and for each subtype there were multiple runs where the number of processors were varied. For all but one subtype of run (Teradata Benchmark), the number of processors was varied from 4 to 1024 by a factor of 2.

In addition, the model was altered to reflect a new software release and the model was re-run where the new release was expected to produce significant changes.

Two different types of ratios of AMPs to IFPs were used: 2:1 and balanced. Since the total number of processors was always a power of 2 (4, 8, 16, ...), it was not always possible to achieve the desired 2:1 ratio. The ratio actually ranged from 1:1 for four processors, 1.67:1 for eight processors to 1.99:1 for 1024 processors. The balanced ratio was obtained by adjusting the ratio of AMPs to IFPs to obtain optimum results for each model run.

Two modes of submitting requests were modeled: Interactive and Batch. The difference in these modes was in the parsing times at the IFPs. Interactive requests used .333 seconds for Primary Retrievals, .667 seconds for Secondary Retrievals, .4 seconds for Updates, and .667 seconds for Joins. Batch Retrievals used .1 seconds for all requests. These numbers were provided in reference manuals [DBC-3].

5.3.1 General DBMS Operations

Four different subtypes of generic operations were modeled:

- o Primary Key Retrieval
- o Secondary Key Retrieval
- o Update
- o *Join*

Values output from model runs and components identified as potential bottlenecks are included in Appendix D of this report.

5.3.1.1 Primary Key Retrieval

A Primary Key Retrieval is a search against a unique primary key where only one AMP is involved in the search. Four varieties of Primary Key Retrievals were modeled:

- o Interactive Retrieval, where the ratio of AMPs to IFPs was 2:1.
- o Interactive Retrieval, where the ratio of AMPs to IFPs was balanced.
- o Batch Retrieval, where the ratio of AMPs to IFPs was 2:1.
- o Batch Retrieval, where the ratio of AMPs to IFPs was balanced.

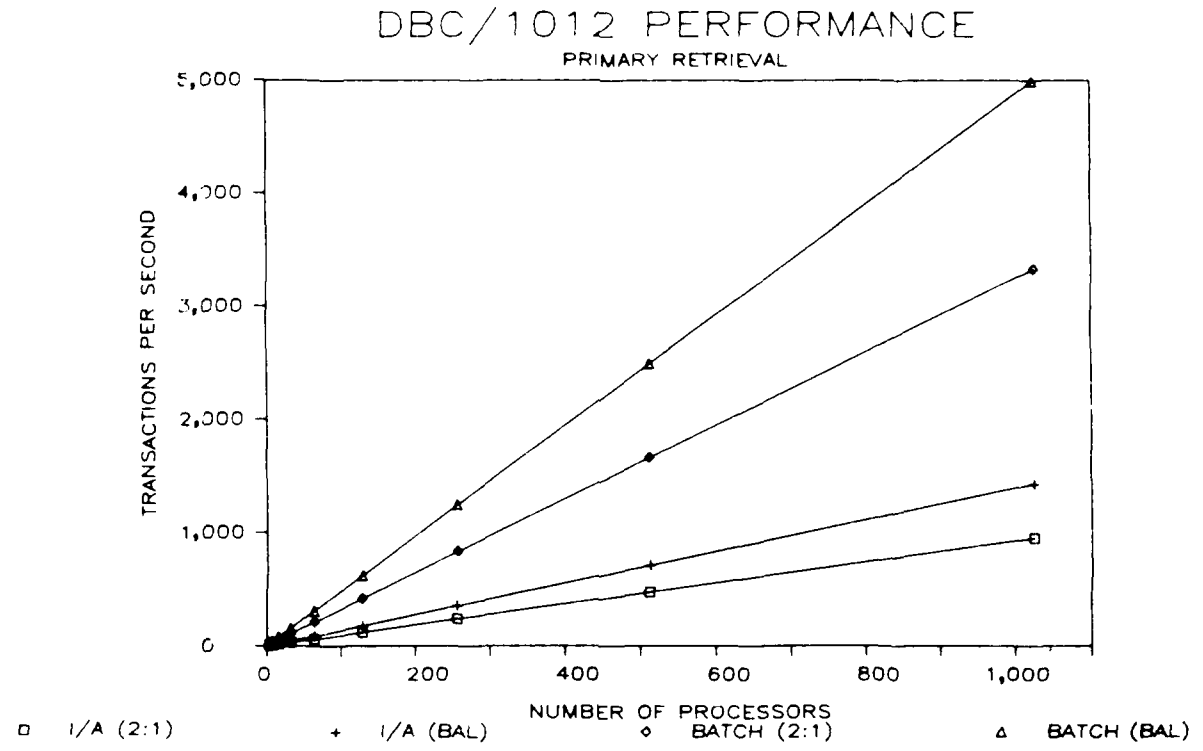
When modeling the Primary Key Retrievals, the number of Primary Retrieval Requests per user per minute was set to one and the other three types of requests were set to zero. The number of users was adjusted for the maximum number of transactions per second that would produce a Primary Request *Response time of five seconds while the IFP, DSU, and AMP RHOs* stayed below .9999 (full utilization).

The number of transactions per second when modeling Interactive Requests varied from 5.58 with 4 processors, to 955.33 with 1024 processors, at the 2:1 ratio; and from 5.58 to 1431.48 transactions, respectively for a balanced ratio. The throughput was somewhat better when modeling batch requests; transactions varied from 19.47 per second with 4 processors (2:1), to 3330 per second with 1024 processors (2:1), and from 19.47 per second with 4 processors (balanced), to 4985.1 per second with 1024 processors (balanced).

The increase in throughput was linear, as predicted. In all cases the IFPs were the bottleneck, and consequently, all balanced ratios of AMPs to IFPs were 1:1. See the upper half of Figure 5-4 for a graph portraying the results of modeled primary retrievals.

5.3.1.2 Secondary Key Retrieval

Two keys were used as the average number for Secondary Key Retrieval in the model. Under the current release of the Teradata software, all AMPs that contain data for the table being searched participated in the request. In the model, all tables were spread over all of the AMPs, so all AMPs were involved in the search. Under the next Teradata software release, secondary keys will be hashed in a manner similar to primary keys so that only those AMPs which contain the data will be involved in the search. This will enhance performance for operations involving simple (equality) searches on unique secondary keys and may enhance non-unique secondary key options. However, non-equality or range search operations are presumed to be unaffected by this new release, since all AMPS must be involved in the search.



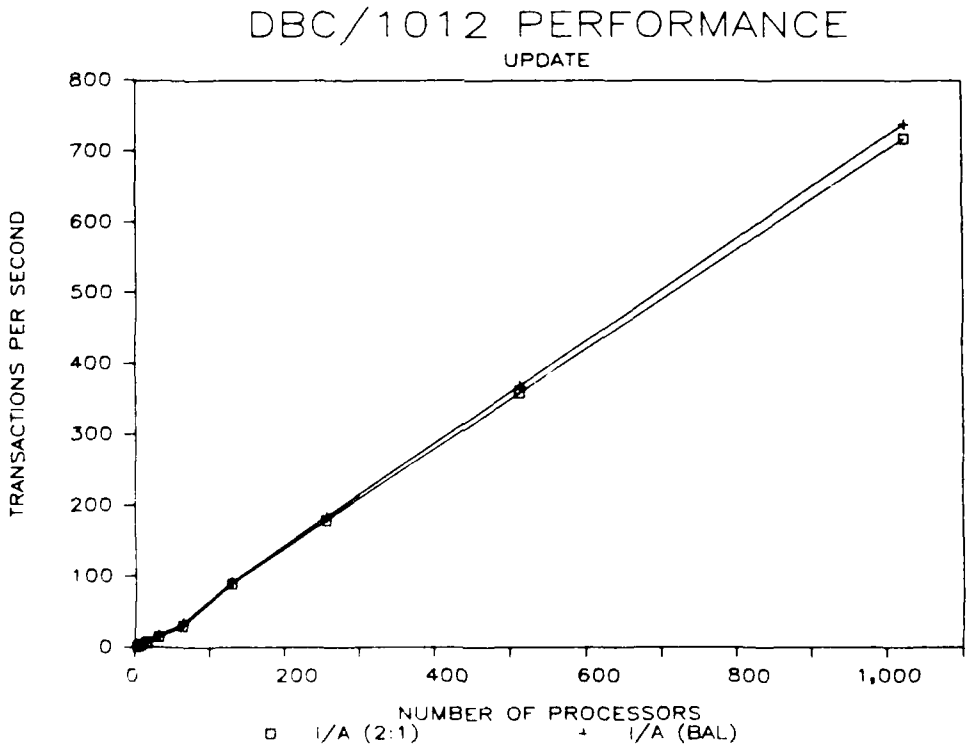


Figure 5-4 Primary Retrieval and Update

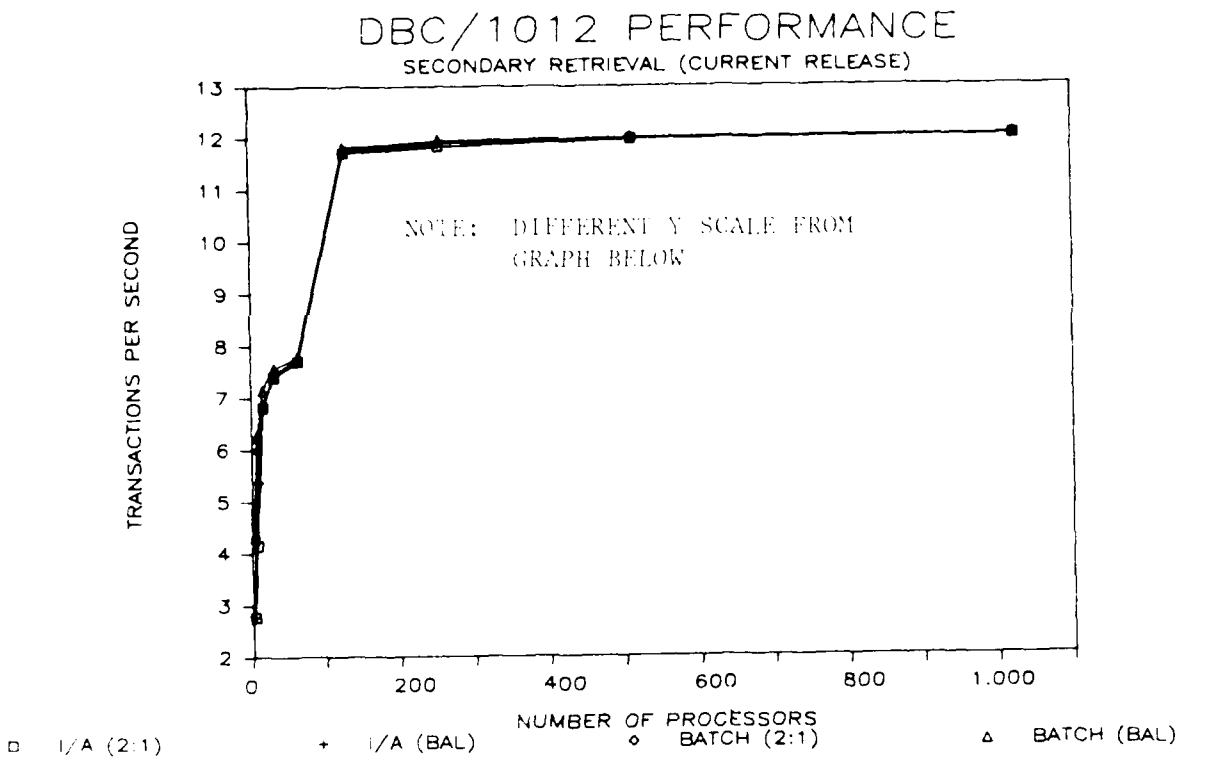
Four varieties of Secondary Retrievals for both the current and next release were modeled:

- o Interactive Retrieval, where the ratio of AMPs to IFPs was 2:1.
- o Interactive Retrieval, where the ratio of AMPs to IFPs was balanced.
- o Batch Retrieval, where the ratio of AMPs to IFPs was 2:1.
- o Batch Retrieval, where the ratio of AMPs to IFPs was balanced.

When modeling Secondary Key Retrievals, the number of secondary retrievals per user per minute was set to one, and the other three types of requests were set to zero. The number of users was adjusted for the maximum number of transactions per second that would produce a secondary request response time of 10 seconds while the IFP, DSU and AMP RHOs stayed below .9999 (full utilization). Under the current release, the four varieties of requests produced near identical results ranging from 2.77 transactions per second with 4 processors to 12.03 transactions per second with 1024 processors.

The DSUs proved to be the bottleneck in all cases except for Interactive Requests when using from 4 to 16 processors. In this case, the IFPs were the bottleneck. When modeling a balanced system, the ratio of AMPs to IFPs rapidly rose to 15:1 since the DSUs were the bottleneck. In all cases, the throughput rises rapidly, and then levels off at 64 processors where it takes a sharp rise to 128 processors where it remains virtually flat. See the upper half of Figure 5-5 for a graph portraying the results of modeled secondary retrievals (current release).

The number of reads per DSU decreases as more AMPs are added to the system. One DSU per AMP was modeled in the runs being made. The average number of reads rapidly decreases from 5.5 to just above 3, which



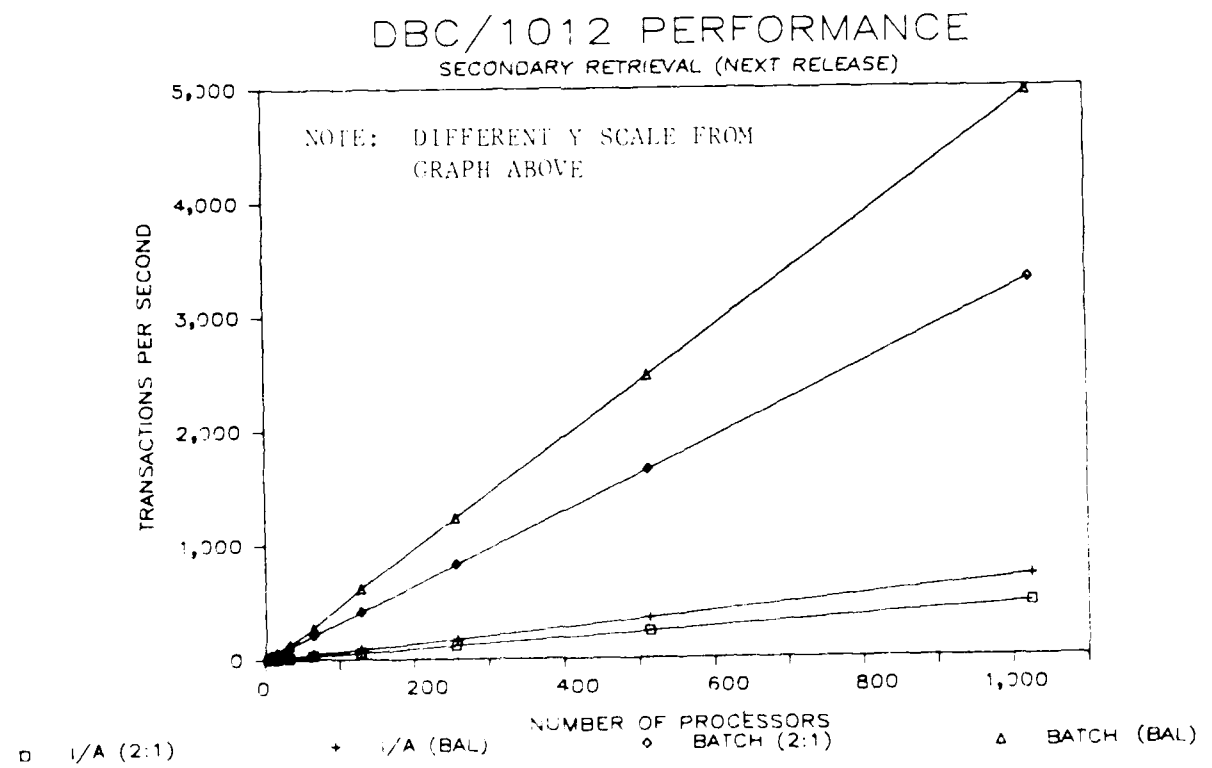


Figure 5-5 Secondary Retrieval

represents the number of index reads. As the table gets spread out over more DSUs, the number of rows per DSU gets small enough that only two levels of the B-tree index are required, and therefore, the number of reads drops from just above three to just above two (average) where it remains throughout the expanded configuration range..

The point at which this sharp rise occurs is governed by four factors:

- o The size of the key plus overhead
- o The size of the index blocks
- o The number of rows in the table
- o The number of DSUs per AMP

Since none of these values were changed in the model runs that were made, the number of AMPs at which this change occurs remained constant at slightly over 61 AMPs. Reference Table 5-1 which shows reads per DSU for both 2:1 and balanced ratios as the number of processors increase. This table shows that the drop from about 3 to about 2 reads occurs in both cases between 64 and 128 processors. This is between 42 and 85 AMPs for the 2:1 ratio, and 60 and 120 AMPs for the balanced ratio. The critical point is at 61+ processors since at 128 keys per block, 16384 Rows per DSU is the maximum size at which only two levels of index are needed. 128 key records per block is arrived at when 3584 bytes per block is divided by 12 bytes key size plus 16 bytes of overhead.

When modeling the next Teradata software release, the Secondary Retrieval throughput was much like the Primary Retrieval with a maximum of 4955.67 transactions per second and linear increase in throughput with increasing number of processors, as indicated in the lower half of Figure 5-5.

2:1 RATIO

Number of Processors	4	8	16	32	64	128	256	512	1024
Number of AMPs	2	5	10	21	42	85	170	341	682
Reads/DSU for Search	3	3	3	3	3	2	2	2	2
Reads/DSU for Retrieval	2.5	1	.5	.24	.12	.06	.03	.01	.007
Total Reads/DSU	5.5	4	3.5	3.24	3.12	2.06	2.03	2.01	2.007

BALANCED RATIO

Number of Processors	4	8	16	32	64	128	256	512	1024
Number of AMPs	2	6	14	31	60	120	240	480	960
Reads/DSU for Search	3	3	3	3	3	2	2	2	2
Reads/DSU for Retrieval	2.5	.83	.36	.16	.08	.04	.02	.01	.005
Total Reads/DSU	5.5	3.83	3.36	3.16	3.08	2.04	2.02	2.01	2.005

Table 5-1 Reads per DSU for Secondary Retrieval (Current Release)

5.3.1.3 Update

An Update of a Row was modeled using a primary key retrieval and modification to all secondary keys in the Row. Two varieties of updates were modeled:

- Interactive Update, where the ratio of AMPs to IFPs was 2:1.
- Interactive Update, where the ratio of AMPs to IFPs was balanced.

When modeling the updates, the number of update requests per user per minute was set to one, and the other three types of requests were set to zero. The number of requests was adjusted for the maximum number of transactions per second that would produce an update response time of 30 seconds, while the IFP, DSU, and AMP RHOs stayed below .9999 (full utilization).

The number of transactions per second varied from 1.43 with 4 processors to 717.70 with 1024 processors when using a 2:1 ratio and a slightly better throughput was obtained when using a balanced ratio. The graph in the lower half of Figure 5-4 portrays the results of modeling updates.

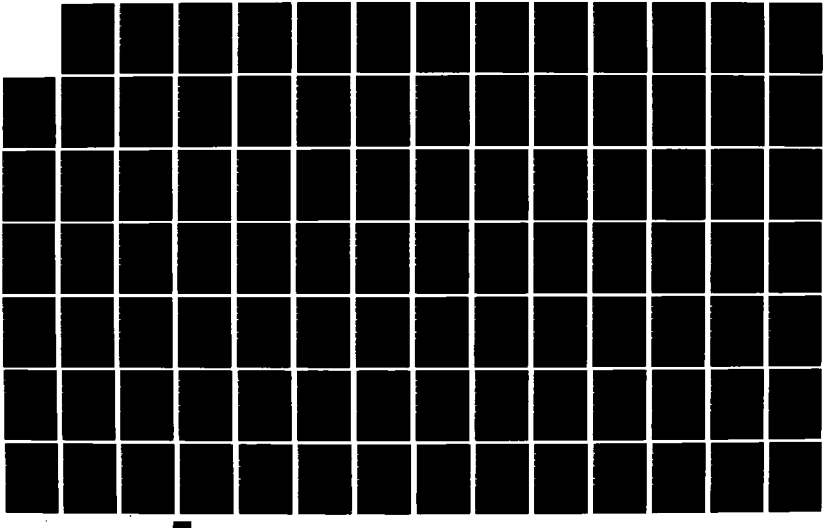
The updates also produced a near linear curve similar to the interactive primary retrievals, but with a slightly lower throughput due to the extra work of rewriting the records.

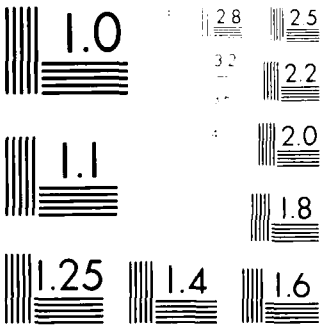
5.3.1.4 JOINs

The JOIN as modeled is a search on a secondary key in one table and a JOIN with a Row Identifier in a second table. Two varieties of JOIN were modeled using both the current release and the next release:

- Interactive JOIN, where the ratio of AMPs to IFPs was 2:1.
- Interactive JOIN, where the ratio of AMPs to IFPs was balanced.

AD-A169 300 C3I (COMMAND CONTROL COMMUNICATIONS AND INTELLIGENCE) 2/3
TERADATA STUDY(U) MEASUREMENT CONCEPT CORP ROME NY
J DECKER MAR 86 RADC-TR-85-273 F30602-85-C-0029
UNCLASSIFIED F/G 17/2 NL




1.0
2.8
2.5
3.2
2.2
1.1
2.0
1.8
1.25
1.4
1.6

5.3.1.3 Update

An Update of a Row was modeled using a primary key retrieval and modification to all secondary keys in the Row. Two varieties of updates were modeled:

- Interactive Update, where the ratio of AMPs to IFPs was 2:1.
- Interactive Update, where the ratio of AMPs to IFPs was balanced.

When modeling the updates, the number of update requests per user per minute was set to one, and the other three types of requests were set to zero. The number of requests was adjusted for the maximum number of transactions per second that would produce an update response time of 30 seconds, while the IFP, DSU, and AMP RHOs stayed below .9999 (full utilization).

The number of transactions per second varied from 1.43 with 4 processors to 717.70 with 1024 processors when using a 2:1 ratio and a slightly better throughput was obtained when using a balanced ratio. The graph in the lower half of Figure 5-4 portrays the results of modeling updates.

The updates also produced a near linear curve similar to the interactive primary retrievals, but with a slightly lower throughput due to the extra work of rewriting the records.

5.3.1.4 JOINs

The JOIN as modeled is a search on a secondary key in one table and a JOIN with a Row Identifier in a second table. Two varieties of JOIN were modeled using both the current release and the next release:

- Interactive JOIN, where the ratio of AMPs to IFPs was 2:1.
- Interactive JOIN, where the ratio of AMPs to IFPs was balanced.

When modeling the JOIN, the number of JOIN requests per user per minute was set to one and the other three types of requests were set to zero. The number of users was adjusted for the maximum number of transactions per second that would produce a Join response time of 45 seconds while the IFP, DSU and AMP RHOs stayed below .9999.

When modeling the current Teradata software release, the throughput achieved for a 1024 processor configuration was rather low: 18.8 transactions per second when using a 2:1 ratio, and 20.12 transactions per second when using a balanced ratio. The curve flattened out due to the use of the Secondary Retrieval.

When modeling the next Teradata software release, the throughput climbed rapidly until between 128 and 256 processors where it suddenly leveled off at just over 100 transactions per second when using both 2:1 and balanced ratios. This leveling off was caused by excessive AMP-AMP communication. This is demonstrated by the AMPs replacing the IFPs as the bottleneck at 128 processors. See Figure 5-6 for graphs of both current release and next release of the modeled performance for Interactive JOIN.

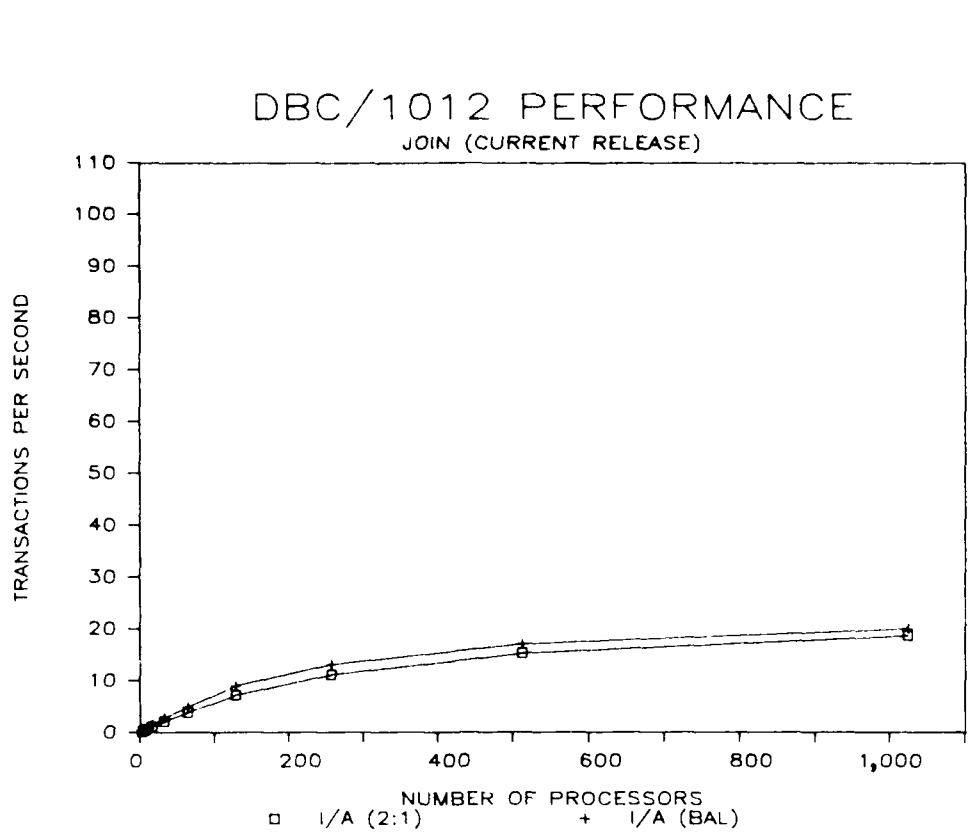
5.3.2 C3I Tests

Two different types of C^3I tests were modeled:

- Photo Interpretation (PI) Mix
- Indications and Warning (I&W) Mix

5.3.2.1 PI Mix

The PI Mix consists of a mix of three of the four generic DBMS requests representing the type of requests a PI Analyst would typically make. This was modeled by .65 Primary, .325 Secondary, .325 Update, and Zero JOIN Requests per user per minute for a total of 1.3 requests per minute based



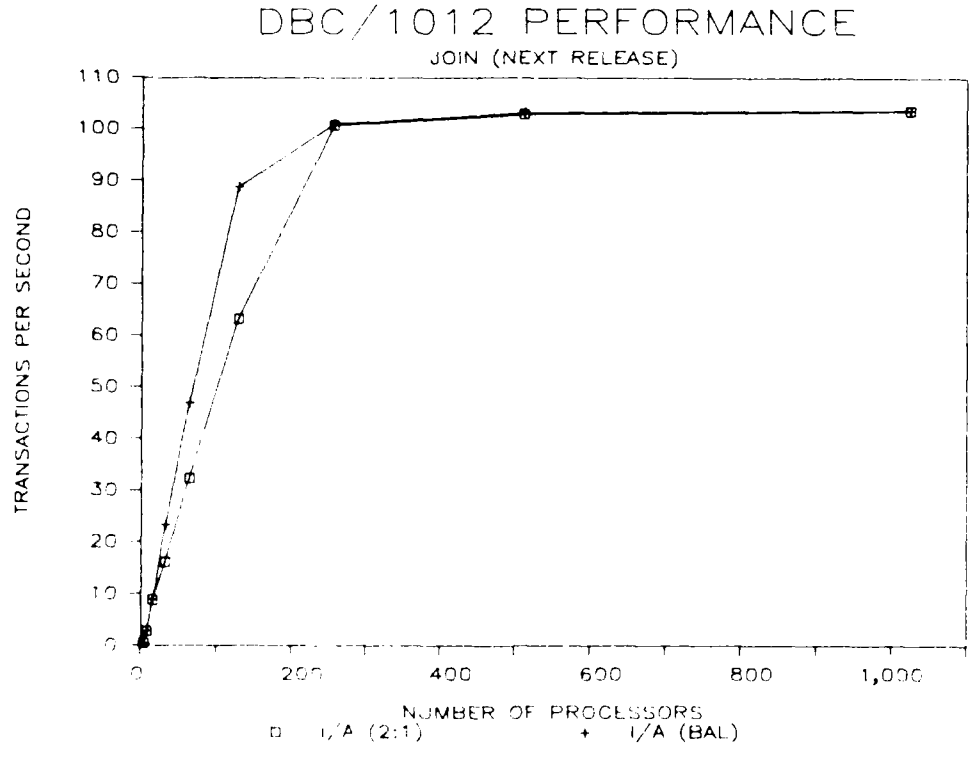


Figure 5-6 JOIN Operations

on predicted 1990 SAC workloads cited in the June 1982, IDHS-80 Evaluation Final Report prepared by Mc^2. The number of users was adjusted for the maximum number of transactions per second that would produce a primary request response time of five seconds while the IFP, DSU and AMP RHOs stayed below .9999.

The number of transactions rose to just above 40 per second at 128 processors for both 2:1 and balanced ratios. The transactions remained virtually flat, climbing only to 47+ at 1024 processors. This was due to the Secondary Retrieval Component of the PI Mix.

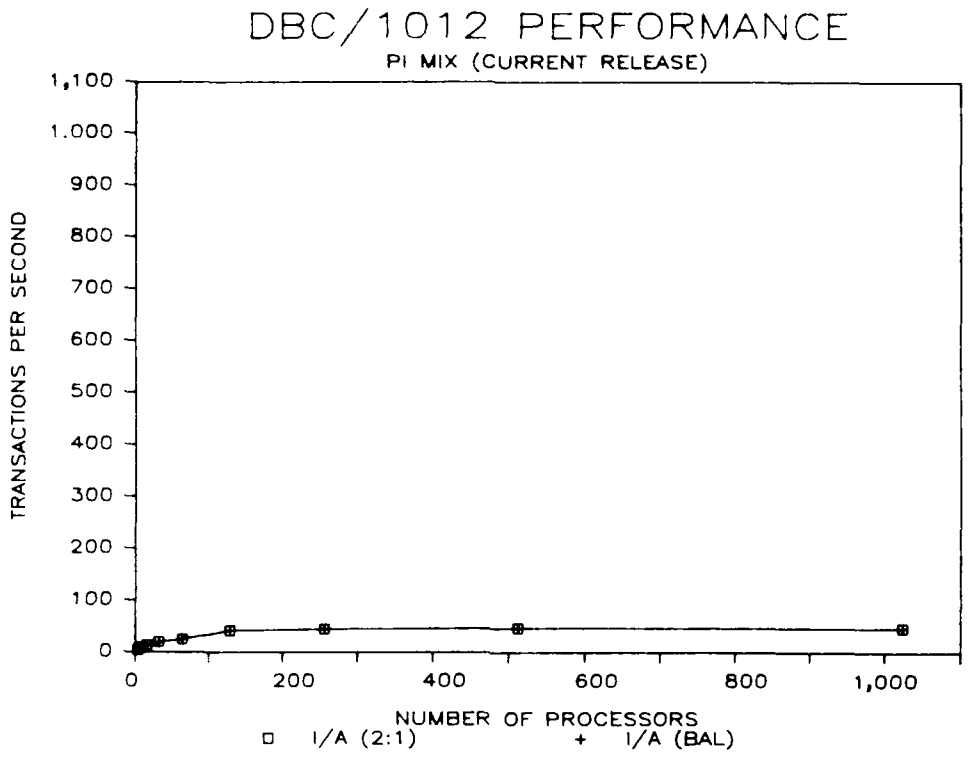
When modeling the next release throughput made a linear increase to 734 transactions per second for the 2:1 ratio at 1024 processors and 1097 transactions per second for the balanced ratio. See Figure 5-7 for both current release and next release of the modeled performance for the PI Mix.

5.3.2.2 I&W Mix

The I&W Mix consists of a mix of all four generic DBMS requests representing the type of requests an I&W analyst would typically make. This was modeled by .5 Primary, .2 Secondary, .1 Update, and .2 JOIN requests per user per minute for a total of one request per user per minute. The number of users was adjusted to produce an average response time of 30 seconds while the IFP, DSU, and AMP RHOs stayed below .9999.

The throughput for the current release I&W model runs was much like that of the PI Mix except that the number of transactions only reached 19 per second at 1024 processors. This was due to the Secondary Retrieval component of the I&W Mix.

There was a large improvement in the I&W throughput when modeling the next release. Both 2:1 and balanced ratios climbed in a near linear fashion, with the balanced being sumewhat higher until 512 processors were reached, at which point both started to level off reaching approximately 475



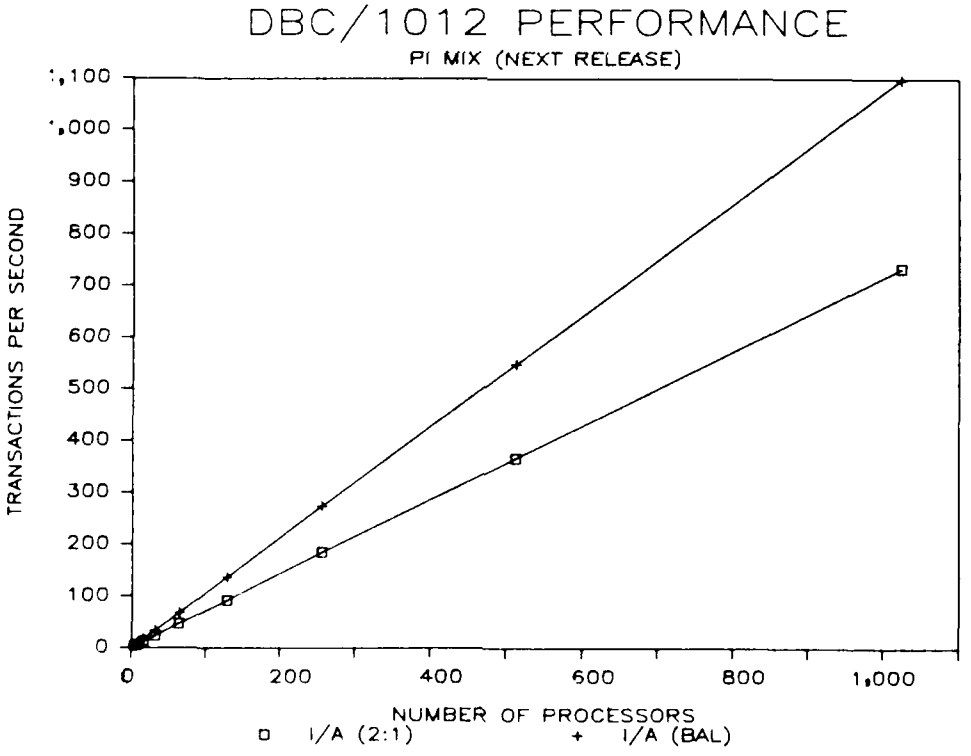


Figure 5-7 Photo Interpretation Operations Mix

transactions per second at 1024 processors. This leveling off is caused by the JOIN component of the I&W Mix, which causes excessive AMP-AMP communication at large numbers of processors. This is shown by the AMP becoming the bottleneck at 1024 processors for the 2:1 ratio and at 512 and 1024 processors for the balanced ratio.

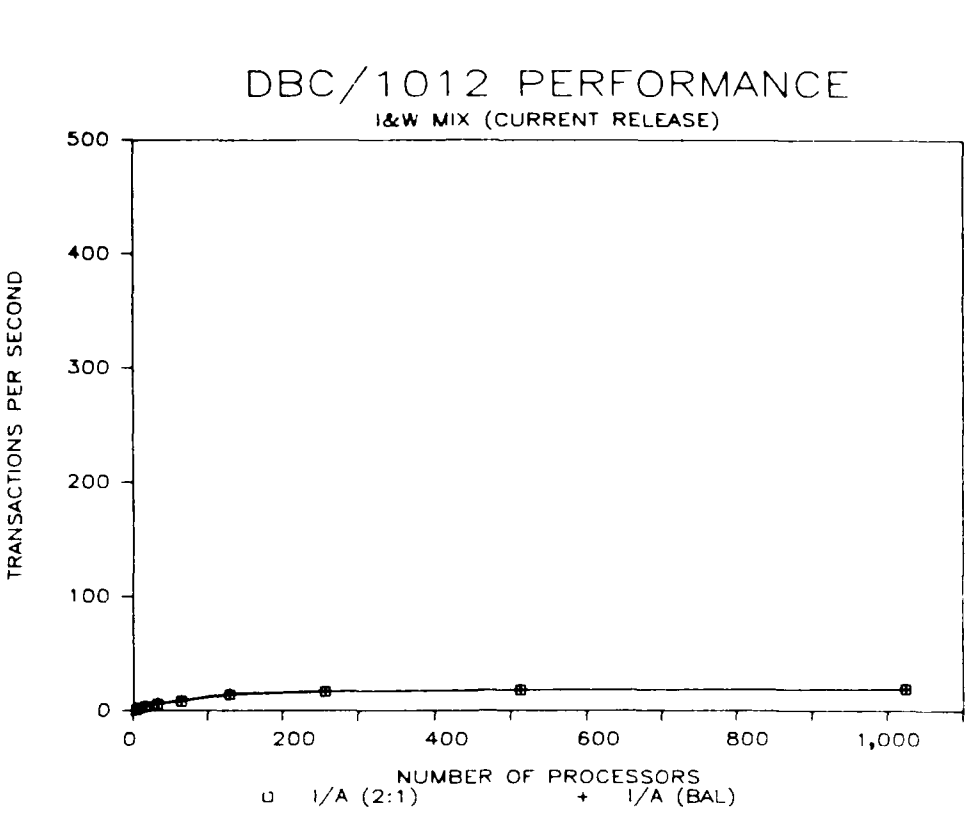
See Figure 5-8 for both current and next Teradata software release of the modeled performance for the I&W Mix.

5.3.3 Teradata Benchmark Tests

Teradata Corporation ran Benchmark tests using 6, 32, and 60 processors. An attempt to model these Benchmarks was made. Some modeling parameters from the Teradata runs were unknown so they were left at the value used in other runs. The table size was known to be 1,000,000 rows and the processer mix of AMPs to IFPs for the 60 processors was known to be 40:20 or 2:1. The number of keys in a row was one instead of two as on other model runs. Four varieties of the Benchmark tests were run:

- Teradata Benchmark
- Teradata Benchmark, Extended
- Teradata Benchmark, Modified

The Benchmark tests consisted of 65% Primary Requests and 35% updates. For the model runs, the number of Primary Requests, Secondary Requests, Updates, and JOINs per user per minute were set to .65, 0, .35, and 0, respectively. The number of users was adjusted for the maximum number of transactions per second that would produce a Primary Key Retrieval Response time of five seconds, while the IFP, DSU, and AMP RHOs stayed below .9999.



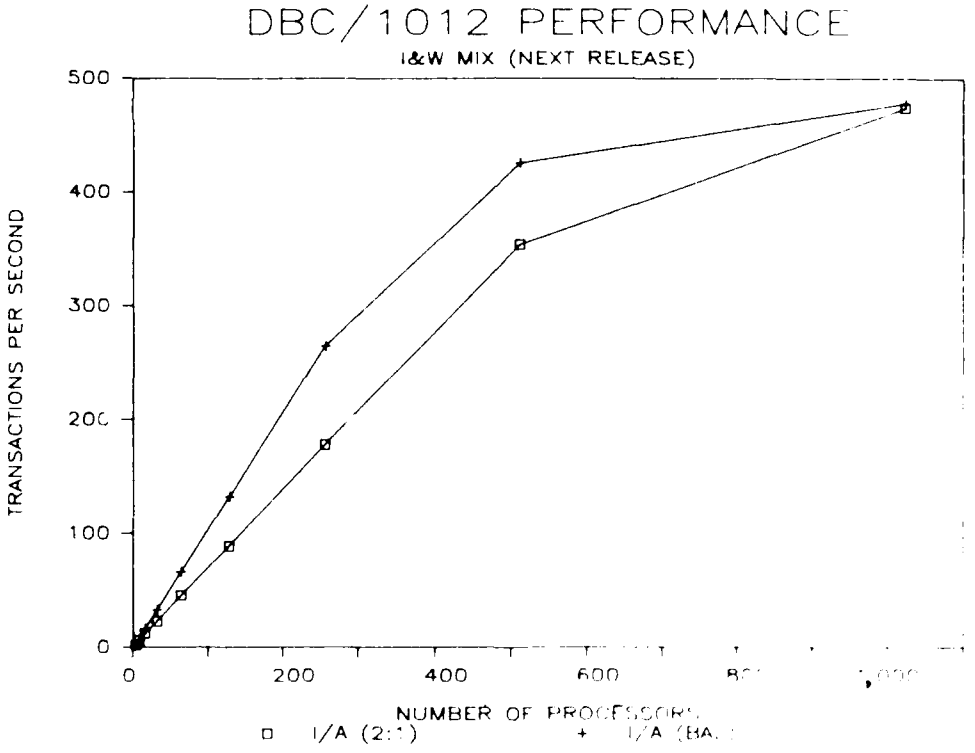


Figure 5-8 Indications and Warning Operations Mix

5.3.3.1 Teradata Benchmark

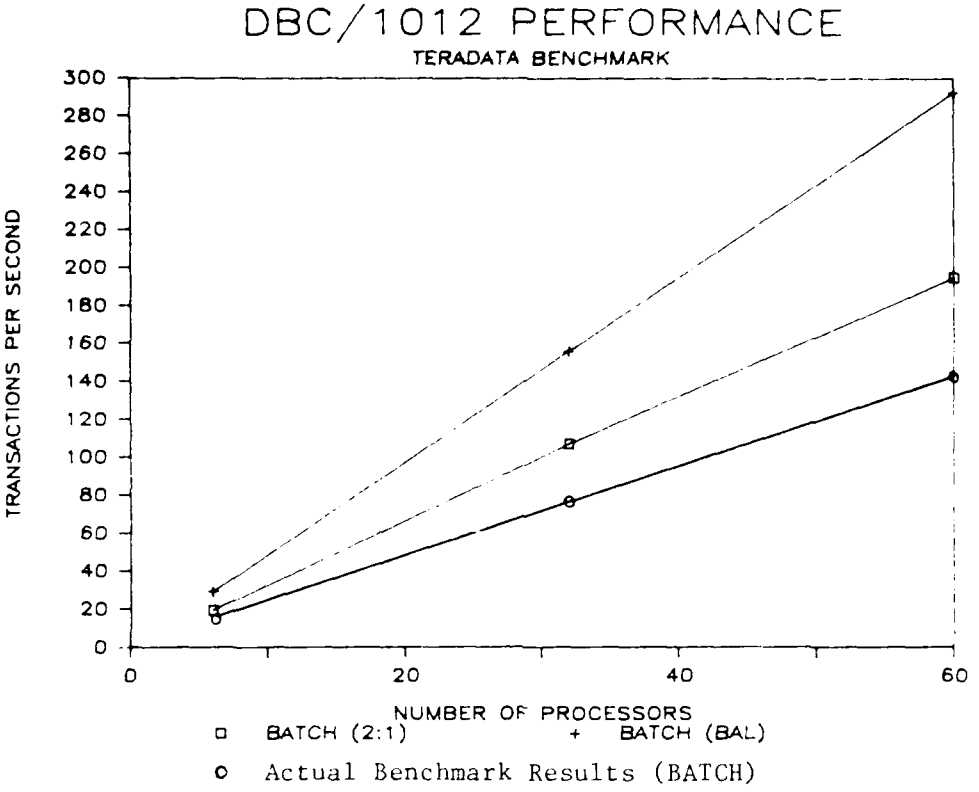
This model run was made to duplicate the actual Benchmark tests using 6, 32, and 60 processors with the 2:1 ratio believed to be used in the actual Benchmark tests. Balanced ratio runs were also made. Comparison between the model outputs and Teradata Benchmark results are as follows:

	NO. PROCESSORS		
Transactions/Second	6	32	60
Teradata Benchmark	15	83	139
2:1 Model Run	19.5	107	195

A linear performance curve was achieved as in the Teradata benchmark (see top graph in Figure 5-9). Discrepancies in the transaction rates output by the model can be attributable to the operational response time requirement parameter set in this run to five seconds. A smaller response time parameter value would lower the transaction rate, yet achieve the same linearity. Response time control is not exercised in benchmark testing, since all responses are accumulated over the benchmark execution period.

5.3.3.2 Teradata Benchmark, Extended

This set of model runs was an extension of the previous runs using the standard numbers of processors (4, 8, 16, 32, ... 1024) instead of 6, 32, 60 processors. This test showed that the number of transactions per second would increase in a linear fashion as the number of processors was increased to 1024. See Figure 5-9 for graphs of the Teradata Benchmark as modified and of the Benchmark extension as modified.



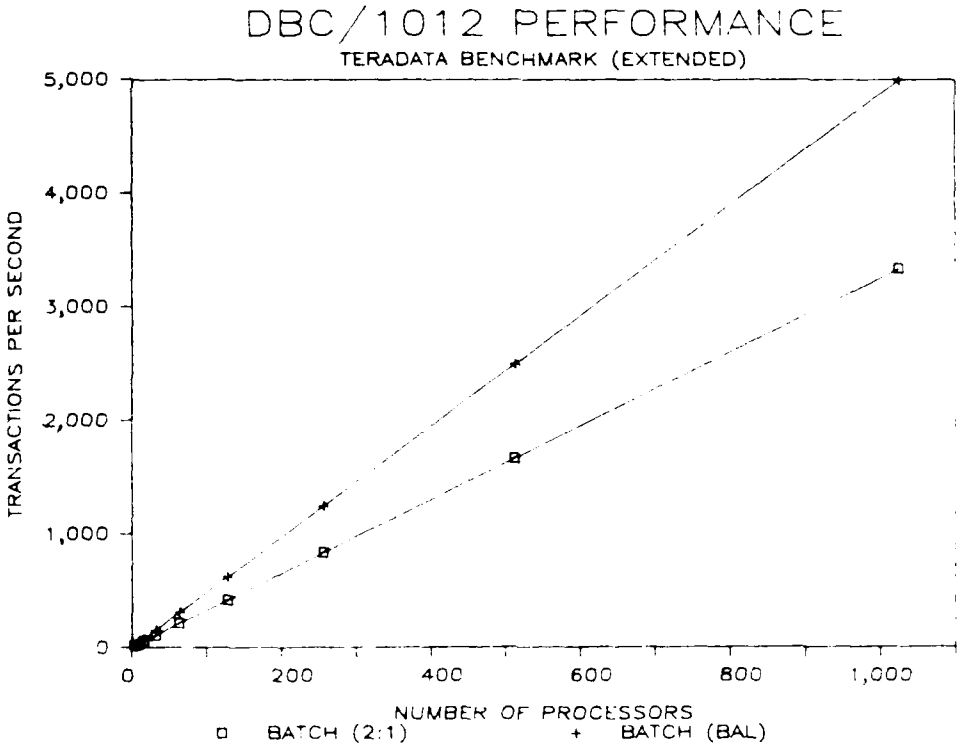


Figure 5-9 Teradata Benchmark and Extended Benchmark Outputs

5.3.3.3 Teradata Benchmark, Modified

The Teradata Benchmark, Modified model runs were the same as the Teradata Benchmark, Extended except the number of keys per row was set to the usual value of 5. This lowered the throughput from a maximum of 3332 to 1970 transactions per second for 1024 processors with a 2:1 ratio. It also lowered throughput for 1024 balanced processors from 4987 to 2255 transactions per second. See Figure 5-10 for a graph of the Teradata Benchmark as modified.

5.4 Conclusions

The Teradata model would seem to indicate that the DBC/1012 will process very large amounts of data, and for some types of requests such as Primary Request and Update, the response is linear, as more processors are added, a corresponding increase in transactions per second is achieved, but for other types such as Secondary Retrieval and JOIN, a point is reached where additional processors will produce little, if any, improvement in throughput. The next release of the software appears to make the secondary request performance curve also linear, but the JOIN still has an upper limit to the throughput.

There was not enough time to exercise the model to its full extent. Some parameters such as key size, number of records in table and index block size were not varied. By changing these parameters, the point at which the throughput flattens out could be moved and possibly produce a near linear response over a wider range of number of processors.

This model was implemented for equality Primary Retrievals. It is our opinion that range searches on primary keys would behave much like the secondary searches behave on the current release.

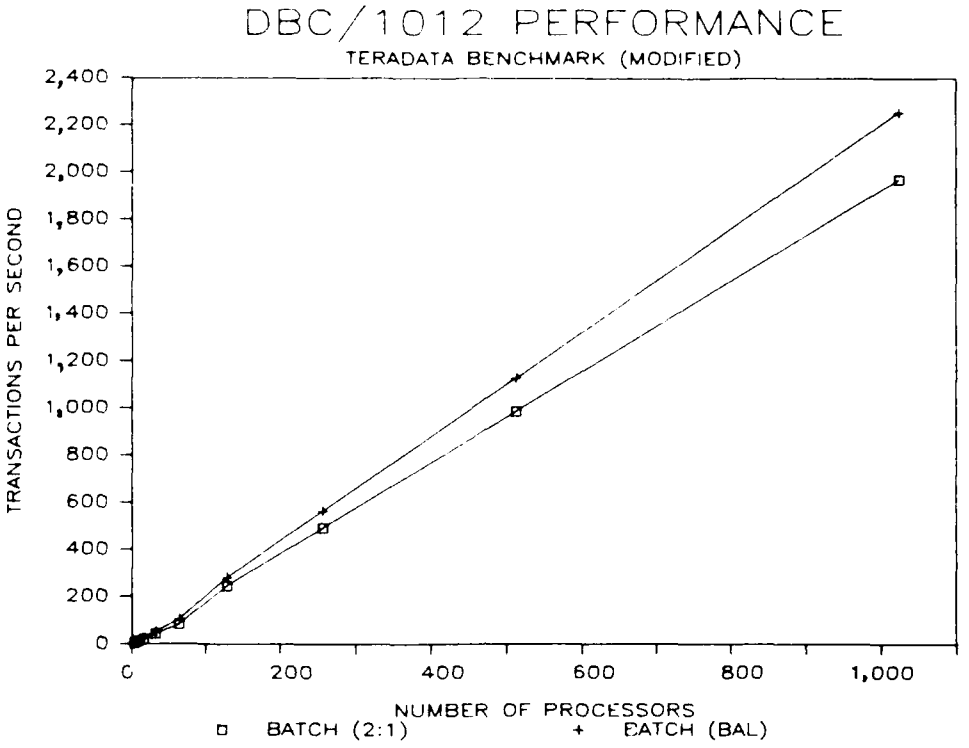


Figure [illegible] Benchmark (Modified)

SECTION 6
C^3I APPLICABILITY ASSESSMENT

This section presents an assessment of the DBC/1012's applicability to the C^3I data management environment in terms of its functionality, integrity, and data base administration support. Approaches for inserting the DBC/1012 into existing C^3I environments are also discussed. (For a more detailed discussion of this topic, refer to the second Interim Report produced under this effort [MCC-2]).

6.1 Functionality

The extent of support that the DBC/1012 can provide to C^3I data processing functions is assessed by analyzing its methods of operation and determining how its features apply to a set of typical C^3I operations.

6.1.1 Methods of Operation

The DBC/1012 supports three general methods of operation:

- Interactive ad hoc query/update under ITEQ, and the capability to build and execute streams of interactive data base requests under BTEQ;
- Interactive menu-based transaction processing, via CICS interface; and
- Applications code driven functions which use CLI or Preprocessor interfaces.

No shortcomings are seen in accessing a session environment suited to the data base processing application.

6.1.2 Applicability to C3I Functions

A flow of generalized C^3I functions was developed to form a basis for the functional applicability analysis. In the following subsections, the functions in this flow and associated support provided by the DBC/1012 are discussed.

6.1.2.1 Accept and Validate Data from Operational Sources

Data from operational sources includes message data arriving electronically over communications systems such as Autodin and IDHSC-II. These messages normally are in a standard report format, such as GENSER, DSSCS, or IDHSC-II. Other data arriving at a C^3I center includes extracts and/or updates of national data bases, such as the Automated Installations File (AIF), and orders of battle (OBs). This information may arrive over electronic media, but is usually on magnetic media, such as magnetic tape volumes.

The DBC/1012 features a "foreign file" capability which reads data files from the host and updates, deletes, or inserts records in the data base. This capability is implemented under the Bulk Data Load (BDL) utility. The input data may be a data set or data records output from a host-resident preprocessor program.

Special data types, such as the GEO COORD definition (found, for instance, in the SARP DBMS) are not provided. This data type would be particularly useful for C^3I applications. We see no major development difficulties in formulating C^3I oriented data types for the DBC/1012. In addition, should a more integrated and supportable capability be desired, we feel that Teradata would consider adding these specialized data types in the interest of expanding the market potential of their product.

Data manipulation statements in the BDL session can be used to update or add to specified records in the data base. For instance;

```
UPDATE Installations:Airfields (RnwayLgh=@NRnwayLgh)
  WHERE AfID = @NAfID AND
        RnwayNm = @NRnwayNm;
```

In the above example, NAfID, NRnwayLgh, and NRnwayNm are fields defined in the input record. They are mapped to the data base fields Afid, RnwayLgh, and RnwayNm, respectively.

BDL also performs input validation based on the field data types specified during the DEFINE operation. Records not matching the field definitions are rejected and flagged. The following statistics are collected during the BDL operation:

- Total number of bytes processed;
- Total number of rows inserted, updated, or deleted;
- Total number of rows rejected;
- Total CPU time taken by the bulk operation; and
- Total wall clock time.

6.1.2.2 Browse Data Base in Area of Responsibility

Browsing is an important function for C^3I analysts. For example, an Indications and Warning (I&W) analyst must peruse large amounts of day-to-day data, such as Foreign Broadcast Information Service (FBIS) reports for events which could reveal a change in a country's foreign or domestic policy and eventually affect U.S. regional interests.

The ad hoc query capability provided by ITEQ and BTEQ is well suited to this function. Analysts can query data base tables on primary or secondary keys (indexes), or non-indexed fields. The "HELP" utility can be invoked to identify the columns and indexes of a table or view. Indexes may be created to expedite the retrieval of records based on one or more (up to 16) search fields commonly used by the analyst. The relational join operation, implemented as a RETRIEVE (expression) WHERE (condition) or nested RETRIEVE request, allows the analyst to develop commonalities from several data base tables; for example:

```
RETRIEVE Personalities:ForMinisters.ALL
WHERE OnTravel.VIP = ForMinister.Name AND
      OnTravel.Loc = "Saudi Arabia"
```

would list all foreign ministers currently on travel to Saudi Arabia. Note that the above query extracts information from separate data bases (the current data base, say "Activities", and "Personalities"), and from tables within each data base.

6.1.2.3 Directed Data Base Searches

Directed data base searches are distinguished from ad hoc queries in that they are conducted to arrive at a finding, or to establish some pattern or trend. These activities may take the form of simple or complex requests which are applied to the data base each reporting period to monitor a uniform perspective of related data base information. Examples might be:

"The Months WHERE
The Number of RORSAT Launches Was Greater Than 5
AND
The Number of Bear Recce Intercepts Over Bermuda Was Greater Than 3."

As shown in the previous subsection, the join operation can be used to provide this information on an ad hoc basis. If the above type of request were a normal access mode of operation for the analyst, the DBC/1012 macro construction facility could be used to build standard request structures. The analyst would invoke the macro using specified parameters. This feature could provide an analyst "toolbox" consisting of macro packages available for facilitating analysis.

The TEQUEL "COMMENT ON" statement may be used to provide short descriptions of the macros, thus facilitating operational documentation and training activities; for example:

```
COMMENT ON MACRO HiSurvAct 'Looks for unusually high surveillance
                           activity -- Takes RORSAT mission count
                           and Bear Recce Intercept Count as
                           arguments';
```

Comments can also be entered for Data bases, Tables, Columns, Fields, Views, and Users. They are stored in the Data Dictionary/Directory and thus are available from session to session.

The CICS interface also provides tools for performing uniform structured queries to the data base. Screen displays which prompt the analyst for inputs can be developed under the host CICS environment. The analyst then queries the DBC/1012 using the screen display specified for the task to be performed. Results from the query are transmitted from the DBC/1012 to the host under the CICS interface, and are subsequently output to the analyst's display terminal.

6.1.2.4 Employ Data in Calculations

This function encompasses the utilization of data base information in mathematical calculations such as those performed in terrain analysis, orbit analysis, correlation, and mission planning. DBC/1012 support to mathematical calculations is discussed under the following three areas.

6.1.2.4.1 Applications Code Interface

TEQUEL statements can be embedded in applications programs as discussed earlier in this section. The degree of difficulty encountered in embedding TEQUEL statements in applications code depends to a large extent on the existence of a TEQUEL Preprocessor for the applications program source language. It is relatively simple to set up DBC/1012 - applications code variable linkages and insert TEQUEL statements in the applications program in a Preprocessor supported applications language. If source code is not supported by a Preprocessor, the TEQUEL statements must be stored, either in arrays or files, then read, buffered, and sent to the DBC/1012 under explicit operations.

Several IDHS installations use PL/1 and can therefore employ the Teradata PL/1 Preprocessor to support mathematical processes. COBOL is not used for calculation-intensive applications, but in some installations it still may be used for report generation. In future DBC/1012 software releases, Preprocessors may be developed for other languages; this would depend upon the demand encountered from the DBC/1012 users and potential customer base.

6.1.2.4.2 TEQUEL Operators

TEQUEL statements can invoke both arithmetic and logical operators. The arithmetic operators (in order of precedence) are:

- Unary Plus and Negation
- Multiplication, Division, and Modulo
- Addition and Subtraction
- Aggregate Operators (MAX, MIN, SUM, COUNT, AVERAGE)

TEQUEL does not contain exponential, logarithmic, or trigonometric operators.

Logical operators included in TEQUEL are:

- o Equal to (or Not Equal To)
- o Greater Than (or Equal To)
- o Less Than (or Equal To)
- o AND/OR
- o NOT

6.1.2.4.3 Precision of Numeric Data Values

Several C^3I applications require high precision calculations. An example of such an application is that of determining the geographic locations of objects located by radar surveillance. Radar coordinates must be converted to earth scale coordinates for the object and, in turn, these values are converted to the map scale coordinate reference system used by the C^3I organization. High precision sensor/strike systems are only as effective as the guidance data they employ in seeking out their targets.

INTEGER data types are represented on the DBC/1012 as 32-bit binary words ranging from -2^{31} to $2^{31}-1$. DECIMAL data types represent packed decimal numbers (m.n) where n is the number of digits (up to 15) and m is the scale value ranging from 0 to n.

The precision of the DBC/1012 floating-point number representation is considered adequate for most C^3I applications. The floating point range is considered to be more than adequate. Floating-point number representation in the DBC/1012 is expressed in the IEEE Standard format. This is a sign/magnitude, 64-bit format which features 15 significant digits of accuracy. The range of IEEE floating-point values ($4x10^{-307}$ to $2x10^{308}$) is wider than that for the IBM host floating-point format. Values may be created and stored within the DBC/1012 that will give error messages when converted for delivery to the host.

6.1.2.5 Create Reports

The end products of C^3I analysis activities are, in many instances, reports, such as Warning Reports, Imagery Interpretation Reports, Force Status Reports, etc. These reports are typified by production standards which may designate both fixed and free-text fields. Some reports require that items must be accumulated under various headings.

Two methods may be employed to perform report generation using DBC/1012 facilities. The first involves using an applications programming language, such as COBOL, to direct the data base query operations and then format the data using COBOL report generation facilities. The other method is via ITEQ or BTEQ which both contain report generation facilities. Detailed information on DBC/1012 report generation facilities can be found in [MCC-1].

Report item titles can be calculated in TEQUEL. For example, the following statement performs an aggregate function (SUM) on NumRnways Installation XXX. The SUM operator becomes part of the title:

```
RETRIEVE SUM Facility.Numrnways WHERE InstallNam = XXX;
```

The returned result is:

```
Sum(NumRnways)
      10
```

The calculation of titles is considered a very useful tool in C^3I applications. For instance, it can facilitate I&W ad hoc reporting. It also simplifies system development for standard reports.

6.1.2.6 Structure Data Base Operations to Changing Requirements

Changing requirements may necessitate either changes in the way information is analyzed or changes in the information structure itself. For example, a methodology for analyzing information may be developed which results in a more effective exploitation of the data base. In this case, the data base does not undergo evolution or change, just the patterns of access. Index and macro creation and modification commands can support this type of activity under the DBC/1012 environment.

The second change, data base reorganization, is supported by several DBC/1012 facilities. These will be explained more fully in Section 6.3 of this report.

6.1.3 Summary

It is our opinion that the DBC/1012 generally provides the functionality required for C^3I applications. The satisfaction of specific functionalities is dependent upon assessing the DBC/1012's capabilities against requirements posed in documentation on the respective C^3I systems for which it may be planned.

6.2 Integrity

The aspects of data base integrity are addressed under three principal topics in this section: constraint enforcement, data consistency, and concurrency control. The facilities of the DBC/1012 which support each aspect of data integrity are discussed in detail.

6.2.1 Constraint Imposition and Enforcement

In its narrowest context, integrity concerns the enforcement of constraints placed on data values under all modification (update, add, delete) of the data base [TF-1:31]. A second type of constraint preserves "functional dependencies" [ULL-1:216]; i.e., enables the definition of data base record keys and enforces their correct use in data base manipulation functions. A DBMS which preserves integrity constraints allows their declaration and enforces them during data manipulation operations.

6.2.1.1 Value Constraints

Three TEQUEL DDL statements declare value constraints; CREATE TABLE, CREATE VIEW, and MODIFY TABLE.

CREATE TABLE is used to define the format and characteristics of a data base table. Column attributes in the table include:

- Data Type - DECIMAL n.m, FLOATING, INTEGER, n BYTES, n CHARACTERS, n DIGITS;
- Value Ranges - (numeric TO numeric);
- Length Limits - UP TO n (BYTES/CHARACTERS);
- Default Control - Mandatory column value on insert/update, or NULL value, or default value; and
- Display Format/Column Header specifications.

In most respects, the TEQUEL DDL supports the declaration of value constraints on data base components. The one area which TEQUEL DDL does not appear to support is legal value lists for character-type attributes.

Without the development of applications software support tools, character data may consist of invalid information. Effects run from the obvious ("Nancy Jones' sex is H") to much more serious consequences. For example, consider the case where updates to SAC's MOB data base arrive which erroneously label the Soviet SS-18 heavy ICBM as an SS-28. At least three major undesirable situations arise. The first (and most serious) is that strategic planning activity consumers are basing their calculations on inaccurate and/or missing data. That is, each SS-18 which has been labeled "SS-28" is, in effect, missing from the data base when SS-18s are retrieved. The second is that the MOB data base is growing inexplicably (since there are no SS-28s, those records are merely taking up space in the data base). Periodic purge processes which remove obsolete data are not aware of SS-28 records. The third is that when SS-28 entries are discovered, their presence is ambiguous. (Is this the information on the new SS-28 ICBM? How did we get so many reports on this new ICBM? What is an SS-28?).

The CREATE VIEW statement builds a relational view or window on data within one or more tables. Both columns and rows from tables can be projected and selected. Constraint definition under the CREATE VIEW statement is similar to the CREATE TABLE statement for newly defined columns in the view. For columns which had already been defined in a table, the CREATE VIEW statement allows the imposition of narrower constraints. For example, suppose a table has a DeptNo column which allows the values 100 - 900. Under the view, this column can be constrained to a subset of the table values, such as 200 - 400. This capability is seen as potentially quite useful for functions such as area analyst input.

The MODIFY TABLE statement allows table owner (or privileged user) to change default values, formats, and title attributes. MODIFY TABLE is not used to change data types for table columns.

6.2.1.2 Functional Dependency Declarations

Functional dependency constraints are established during and subsequent to table creation. These are, in essence, the primary and secondary indexes used to facilitate retrieval of table data. Teradata refers to an index as a key defined on values in one or more columns of a table [DBC-4:7-5]. Any table having two or more fields must have a primary index, otherwise an error message is returned [DBC-11:C-236]. Primary indexes may be declared unique. An error message will be generated if duplicate values for a unique primary index are found (for example, during a Bulk Load operation). Only one primary index may be declared for each table.

Up to sixteen secondary indexes may be declared, either in the CREATE TABLE session, or later, via the CREATE INDEX statement. Secondary indexes may be unique or non-unique. If they are defined as unique, duplicate occurrences will be flagged as errors. Unlike the primary index, secondary indexes may be added, dropped, or modified.

6.2.1.3 Constraint Enforcement and Error Reporting

Error detection and reporting occurs both during the constraint definition and insert/update operations. Constraint definition takes place at the table and column levels, while enforcement includes table, column, and also row levels. Constraint definition error messages are generated for both value constraints and functional dependencies.

6.2.2 Data Consistency

The DBC/1012's general rules to ensure consistent data are:

1) Modified data must be committed, i.e., End Transaction statement processed or rolled back.

2) An application cannot access data that has been updated by another application and has not yet been committed or rolled back. This rule can be overridden by the user's use of the Access lock.

3) All updated rows are locked for the exclusive use of the updating application (assuming another user isn't accessing the data via the Access lock), and remain locked until the transaction is committed or rolled back.

4) When commit or rollback occurs, all locks are released.

The implementation of these rules on the DBC/1012 will be discussed in detail in the following paragraphs.

6.2.2.1 Transactions and Commitment

The DBC/1012 employs the Transaction Model and various synchronization techniques to ensure consistency of data. A transaction is a sequence of operations on one or more data base objects (files, records, etc.) that transforms a current consistent state of the system into a new consistent state. When a valid transaction completes successfully, the new state of the system is, by this definition, consistent [KOH-1:151].

A transaction is considered a single unit of work by the DBC/1012. It can be either a single TEQUEL statement or a series of statements which are treated as a single unit. Transactions may be explicitly defined by the BEGIN TRANSACTION (BT) and END TRANSACTION (ET) delimiters, or may be implicitly defined as in certain system utilities, such as DUMP and RESTORE. If a statement in a transaction cannot be completed successfully, the effects of all prior statements in the transaction are nullified. A failure response is returned to the user and the user must bypass execution of the subsequent statements in the transaction. In an explicit transaction, if a statement cannot be completed successfully, a failure response is returned to the user and the entire request is nullified [DBC-11:5-14].

The DBC/1012 logs data manipulation events of a transaction in a temporary Transient Journal. If a transaction is completed, it becomes committed and the data base is updated by the Transient Journal events. If the transaction fails, the original data base entries are restored to their prior state. This policy is an implementation of the "two-phase commit" model [BG-1:190], where Phase 1 involves logging the transaction-produced update commands into the Transient Journal and Phase 2 involves the Transient Journal outputs to the DBMS and any error processing rollback to the original consistent DBMS state.

6.2.3 Concurrency Control

The DBC/1012 employs a locking scheme to serialize transactions. To guarantee data base consistency, all transactions must be well-formed and Two-Phase. A transaction is well-formed if it:

(1) Locks an object before accessing it,
(2) Does not lock an object which is already locked, and
(3) Before it completes, unlocks each object it locked [KOH-1].

A transaction is Two-Phase if no object is unlocked before all objects which will be locked during the transaction are locked. (Two-Phase in this discussion is different from the Two-Phase Commit policy, discussed in Section 6.2.2.1. For clarity, the abbreviation 2PL - for Two-Phase Lock [BG-1] will be used).

DBC/1012 transactions can be considered as either user generated or system generated. System generated transactions can be further categorized as single or multiple statement transactions. User transactions are characterized by multiple statements delimited by a BT and an ET request. System generated transactions are generated when a single request that is not part of a user generated transaction, is received from the host. This request may be treated as either a single statement or a multiple

statement transaction by the system. The primary difference between user generated and system generated transactions is that the system can order the locks of a system generated transaction to minimize deadlocks, but cannot do this for a user generated transaction [DBC-12].

6.2.3.1 User and Host Utility Locks

The DBC/1012 has two categories of locks: user and host utility locks. (The term "user lock" is used here instead of the Teradata-termed "data base lock", which refers to row, table, and data base level locks.) Most user locks are automatically placed by the system during row operations. These row locks are placed on the row hash code portion of the row ID. A row identifier consists of a 4 byte row hash code and a 4 byte row uniqueness identifier. The row uniqueness identifier differentiates rows which have the same row hash ID, for example, for rows which have the same non-unique primary index, or rows which happen to hash into the same 4 byte value. Since the row lock is placed on the hash code portion only, it is possible that more than one row would be locked even though only one row is being accessed by the request. Such a locking policy would appear to have performance implications and would seem to allow the possibility of downright interference among users as opposed the normal contention between competing system processes found on multiuser systems. However, in our discussions with DBC/1012 users, no such performance degradation has been experienced.

The user can override the automatic row lock by using the LOCKING phrase in a TEQUEL statement to lock a data base or a table (row locks cannot be explicitly requested). A table lock will lock all subtables associated with the table, such as indexes and fallbacks. A data base lock locks the data base and all associated tables and subtables.

The user has no control over the release of user locks; these are automatically released by the system when a transaction ends or is aborted.

Host utility locks perform the same functions as user locks but are processed differently by the system. Host utility locks are almost always at the user data base level, except during a DBC (the root data base) dump or restore operation. During a DBC dump operation, the host utility applies a Read lock on tables. During a DBC restore operation, an Exclusive lock is placed on data base tables. Host utilities do not use the Write or Access locks [DBC-12]. (Information on Read, Write, Access, and Exclusive lock types is found in [MCC-1].)

Unlike the user locks, host utility locks are not automatically released at End Transaction time. They remain in force until explicitly released by the user requesting the host utility. The reason for keeping these locks in force is to preserve data base integrity. For example, if a user is dumping a data base and the DBC crashes, the dump must continue upon recovery without any intervening modifications made to the data base. During a restore operation, the Exclusive lock denies any access or modification until the user is satisfied that data base restoration is complete.

6.2.3.2 Lock Compatibility Among Concurrent Users

All accesses to a row require one of the four modes of locks (Read, Write, Access, or Exclusive) to be set on the appropriate object [DBC-12]. The lock compatibility matrix for the DBC/1012 is shown in Figure 6-1. If a lock cannot be granted because of a conflicting lock currently in force, the lock request is queued until the conflicting lock is released, either by the system or the user (in the case of a host utility lock).

The Access lock, as indicated in Figure 6-1, is compatible with the Write lock. The Access lock is used primarily for browsing and can result in the reading of invalid data, if the requested data area is being concurrently written to by someone owning the Write lock.

LOCK NEEDED	CURRENT LOCK BEING HELD				
	NONE	ACCESS	READ	WRITE	EXCLUSIVE
ACCESS	LOCK GRANTED	LOCK GRANTED	LOCK GRANTED	LOCK GRANTED	REQUEST QUEUED
READ	LOCK GRANTED	LOCK GRANTED	LOCK GRANTED	REQUEST QUEUED	REQUEST QUEUED
WRITE	LOCK GRANTED	LOCK GRANTED	REQUEST QUEUED	REQUEST QUEUED	REQUEST QUEUED
EXCLUSIVE	LOCK GRANTED	REQUEST QUEUED	REQUEST QUEUED	REQUEST QUEUED	REQUEST QUEUED

Figure 6-1 Locking Compatibility Matrix

Locks are upgraded by the system when necessary during the processing of requests in a transaction. Most upgrades are from Read locks to Write locks, for example, during a RETRIEVE and subsequent UPDATE on the same row. If another transaction also has a Read lock on the same row, the transaction needing the upgraded lock must wait for that transaction to finish. If two transactions holding a Read lock both need an upgrade to Write lock, a deadlock occurs [DBC-12]. For example, assume that Transactions X and Y hold Read locks on the same row hash. If X needs an upgrade to a Write lock, it must wait for Y to relinquish the Read lock, which is done only at ET. If Y also needs a Write lock upgrade, it must wait for X to finish. X cannot finish, since it is waiting for the upgrade to be granted, and Y cannot finish, because it is waiting for X to end. In the classical deadlock situation, two or more processes are in irreversable contention for two or more data entities. This situation is normally a rare occurrence. In the deadlock type introduced by the DBC, two or more processes are in irreversable contention for <u>one</u> data item, intuitively a higher probability occurrence. Additionally, the hash code, multiple row locking policy multiplies this probability.

Follow up data collection and analysis conducted in this area revealed that no significant deadlock problems and associated degradation have been experienced by customers contacted by Mc2.

Lock upgrades have higher priority than waiting locks in order to minimize deadlock. For example, if Transaction V has a Read lock on a row and requires an upgrade to Write lock, and Transaction W requests a Write lock, the upgrade will be granted. W will not acquire the Write lock until V has ended.

6.2.3.3 <u>Deadlock Detection and Resolution</u>

The DBC/1012 detects deadlocks caused by conflicting concurrent transactions and resolves them by aborting the newest transaction in the conflict. Two types of deadlock situation exist: local deadlocks, which

involve conflicts on a single AMP; and global deadlocks, in which several AMPs are involved in the conflict.

Local deadlocks are detected by an AMP-resident task which is activated approximately every 30 seconds. This task checks the AMP's lock tables for local deadlocks. When a deadlock is detected, the youngest transaction is aborted and rolled back in an attempt to break the deadlock.

Global deadlocks require more information than that available from a single AMP lock table. The Dispatcher in the lowest numbered IFP is assigned the job of coordinating the global deadlock detection function. The Coordinating Dispatcher activates a task every four minutes which broadcasts a message to the Dispatchers in the other IFPs requesting information on any transactions which might be hung up. Criteria for possible hung transactions include:

- Prime key steps for which an AMP response has not been received in two minutes; and

- Any other step for which an AMP response has not been received in four minutes.

A list of transactions meeting these criteria is forwarded from the Dispatchers to the Coordinating Dispatcher. This list is then broadcast to all AMPs as a message requesting that these transactions be checked out. Each AMP checks the status of the requested transactions and a control AMP coordinates a single response for each transaction. Each AMP checks for local deadlock and supplies information to the control AMP which checks for global deadlock. The response sent to the Coordinating

Dispatcher contains the following information for each transaction on the list:

- Transaction is blocked; i.e., waiting for a lock on at least one AMP;
- Transaction is deadlocked; should be a candidate for abort;
- Transaction is working; i.e., still making progress on at least one AMP; or
- Transaction is not blocked, deadlocked, or working; possibly a system error or undetected deadlock.

The Coordinating Dispatcher responds as follows:

- If the transaction is blocked or working, nothing is done, since the transaction is not considered to be in a problem state.
- If the transaction is deadlocked, the youngest of all the transactions is aborted by the Coordinating Dispatcher.
- If the transaction is neither blocked, working, or deadlocked, the Coordinating Dispatcher checks its Ynet output queue to see if the AMP step is unable to be sent to the AMP because of flow control problems. If so, the Coordinating Dispatcher does nothing, assuming that the AMP will be able to accept the step when its processing load lessens. If the step is not in the IFP's output queue, the Coordinating Dispatcher makes an entry in the Error Log and aborts the transaction (it assumes the transaction is hung for some unknown reason). The abort process in this case takes a total of 8 minutes.

6.2.4 Summary

In Section 6.2.2.1, it was shown that the DBC/1012 followed the policy of Two-Phase Commitment, thereby ensuring that data base updates result in consistent data base states. Section 6.2.2.2 stated criteria guaranteeing consistent data in a concurrent transaction environment. A review of how these criteria are satisfied by the DBC/1012 is summarized below.

Are DBC/1012 transactions well-formed?

- Do transactions lock an object before using it? Yes; either automatically, as in row locking by the system, or via explicit locking requests which preceed TEQUEL requests.

- Will a transaction lock an object which has already been locked by another transaction? Only if the locks are compatible as indicated in Figure 6-1.

- Will a transaction unlock its locks before it completes? All locks are released once the transaction has committed or is aborted. The only exceptions are Host Utility locks, which are explicitly released by the utility requester.

Are transactions 2PL on the DBC/1012? According to the Teradata technical documentation [DBC-12] locks are held for the duration of the transaction in which the lock is participating. This satisfies the 2PL criteria. Thus, all data base operations guarantee consistent data base states.

It is our opinion that the DBC/1012 possesses rigorous integrity controls sufficient for C^3I applications.

There are, however, circumstances in which the access, update, and/or utilization of invalid and/or inconsistent data may occur. Recommendations for assuring data integrity are as follows:

- The lack of an "Acceptable Names/Strings Set" DDL construct and enforcement policy can be dealt with via applications software.

- The Access lock should be used only by supervisory or systems support personnel. The temptation to use the Access lock rather than wait until a Read lock is granted must be resisted. Training courses and standard operating procedure and enforcement can support judicious Access lock use policy.

6.3 Data Base Administration Support Assessment

Date Base Administration is generally the responsibility of the Data Base Administrator (DBA). By tradition, this individual is "responsible for the design, creation, integrity, efficiency, and administrative functions of a multi-user data base and all of the files within the data base" [AFA-1:10]. The role of the DBA also includes maintenance of the system when performance is jeopardized and recovery after hardware and software failure.

The design of the data base involves development of the following: data elements, data use identifiers, data dictionary/directory, data structures and storage structures. The data base creation function encompasses creating, initializing, loading, and reorganizing operations. Once created, the integrity of the data base must be maintained and guaranteed. This function includes protection from unauthorized use (security), recovery of data in case of system failure, and control of the data. Another key responsibility of the DBA is to optimize data base efficiency.

Efficiency is evaluated by monitoring performance measures such as mean response time, cost of operation, and storage usage. The major efficiency consideration depends on the system resource for which optimum performance is required. Many of the functions of the DBA are carried out by operators or users under the authority of the DBA.

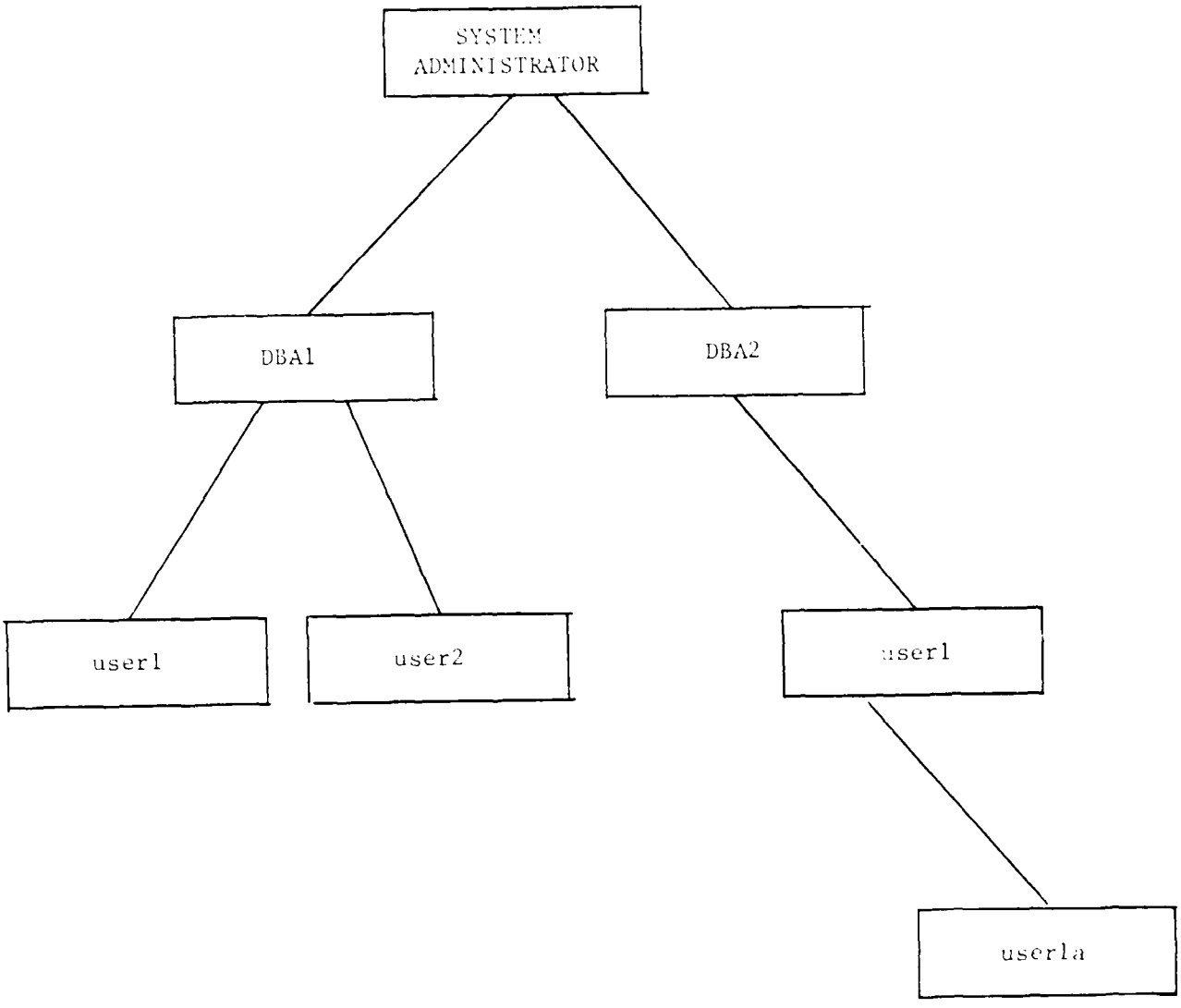
A description of the data base administration tools provided by the DBC/1012 and an analysis of their enhancement to DBA operations follows.

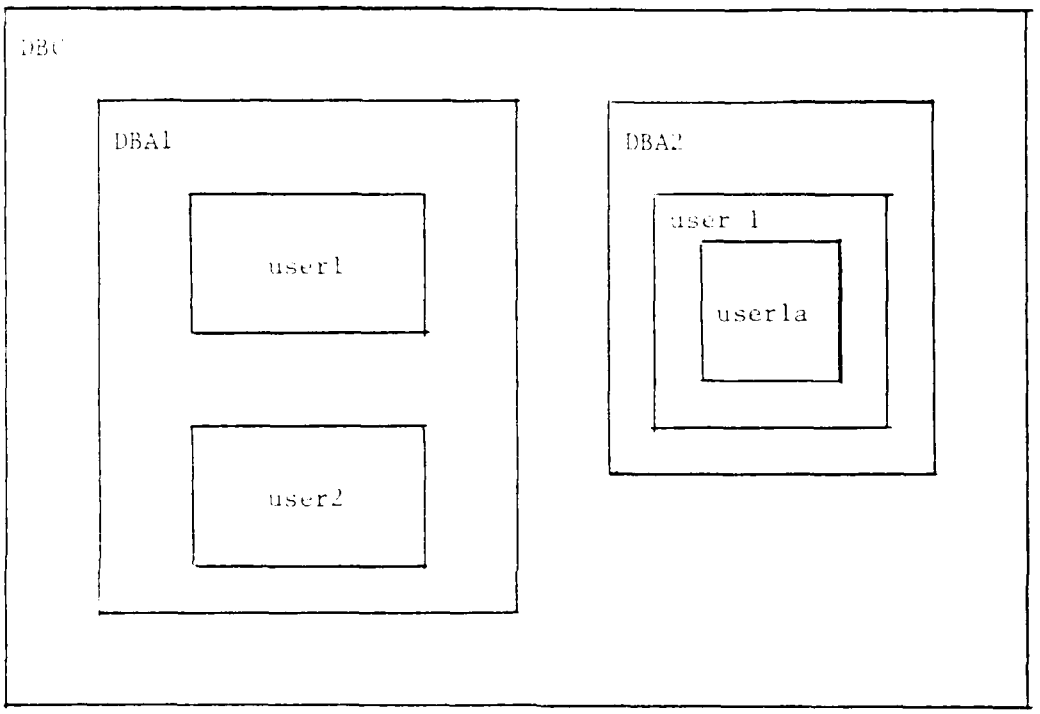
6.1.1 Security Tools

Aspects of the DBC/1012 which address data base security include: organization of the DBC/1012, partition of disk space, privileges accorded users, and information in the Data Dictionary/Directory.

DBC/1012 data base administration authority and control is organized in a hierarchical structure. When the DBC/1012 is initially installed, a DBC/1012 System Administrator is appointed. The System Administrator has control over all DBC/1012 resources. The System Administrator allocates space for system tables and creates a "DBC/1012 Administrator Data Base" from "DBC" (the root data base). Remaining disk space under DBC is then allocated to smaller data bases assigned to various operations of the facility. Each data base has a "supervisory user" who acts as the DBA for that segment of the total data base. When a new data base is created, disk space is allocated from space available in the owner's partition.

Figure 6-2 illustrates the hierarchical organization of the DBC/1012. Figure 6-3 depicts how space is partitioned to owners and how storage space is allocated on the DBC/1012. The hierarchical ownership corresponds to the partitioning of space on the DBC/1012. The figures illustrate examples of how space allocation, control, and access rights can be assigned to DBAs and, in turn, specified supervisory/end users.


SYSTEM
ADMINISTRATOR
DBA1
DBA2
user1
user2
user1
user1a


DBC
DBA1
user1
user2
DBA2
user 1
user1a

Privileges may be granted or revoked by data base owners through the use of TEQUEL statements. Privileges include CREATE, MODIFY, RETRIEVE, UPDATE, DROP, and EXECUTE. The granting of rights may also be delegated. Full delegation of the database is possible via the GIVE statement. (A detailed description of the privileges is contained in the Capabilities and Characteristics Manual [MCC-1]).

DBC/1012 data base security measures are comparable to those featured in most commercial DBMSs. Privileges currently in effect for a user who has successfully signed on the system are determined by information accessed from user profile tables. Before reaching this privilege verification process, a user must have successfully logged on to both the host system and the DBC/1012; a sequence which requires two passwords.

The System Administrator can access all information on users and data bases by examining the Data Dictionary/Directory. The DBC/1012 provides a verification facility that determines whether a user has the privilege to examine information in the Data Dictionary/Directory. This time-consuming access verification process can be bypassed if alternate views, provided by Teradata, are installed on the system. Privileges to use alternate views are granted by the DBC/1012 System Administrator. The granting of alternate view use must be done with extreme descretion, in order to avoid a potential compromise of security.

Whereas the DBC/1012 has a large set of security tools, these tools are, in themselves not sufficient to meet the security requirements of C^3I applications. Although the ability exists to establish access privileges in any desired combination, and at all levels of granularity down to, but not including row level, there is no provision for enforcing higher level security policies based on allowable combinations of these privileges. For example, the Bell and LaPadula "*-property" rule of security, paraphrased, states that a user with TOP SECRET privileges may read any data in the system with TOP SECRET or lower classification, may write TOP SECRET data, but may not write to data files of lower classification (RUS-1 and HAR-1).

Only a user who is logged on at the SECRET level can write to a SECRET file. This seeming contradiction of intent, that a SECRET rated user has privileges which the TOP SECRET user doesn't have, is resolved when one contemplates a scenario in which the TOP SECRET user reads TOP SECRET data and then, inadvertantly or on purpose, writes it to a SECRET file. Unless software is written for the host processor which enforces such policies, the onus is on the DBA to judiciously grant privileges in such a fashion as to not compromise the security of data. Moreover, this is an area of data base administration which cannot be delegated to lower levels in the DBA hierarchy without introducing increased danger of compromise.

At the implementation level, a potential problem exists in certification of system security for DoD purposes. Since the DBC/1012 must be looked at for security purposes as a black box within which its own security constraints are enforced, it must be determined that these constraints, upon which a certifiable security policy must be built, are rigorously and faultlessly executed. Within the complex architecture of the DBC/1012, all data of all security levels is, by the nature of the machine implementation, intermingled, if not necessarily at the IFP or AMP/DSU level, certainly within the Ynet. Questions must be answered as to the likelihood of data exiting the DBC erroneously. Even ignoring the potential non-access to company-proprietary implementation details, it is obvious that the complex nature of the DBC will make it very difficult to "prove" secure.

At the physical level, the DBC/1012 has not been TEMPEST engineered. We see no characteristics of machine packaging which would make it more difficult to shield than other machines. If there is further interest, an estimate should be solicited from Teradata Corporation for TEMPEST shielding.

6.3.2 Recovery Tools

The principal recovery tools implemented on the DBC/1012 are database archive and restore, and table rebuilding utilities.

A data base can be dumped or restored using the utility programs DUMP DATABASE, and RESTORE DATABASE. The DUMP utility archives a copy of the data base from the DBC/1012 to the host data set. The RESTORE utility, restores a data base on the DBC/1012 from an archive data set on the host computer. The RESTORE function can be used to restore data only to the original DBC data base from which it came. Provisions are made for AMPs that may fail during a dump or restore through single-AMP dump/restore routines.

A significant shortcoming of the DBC/1012 is the lack of a "rollforward" utility. There is no facility for logging day-to-day data base operations. After a crash, the system can be restored only to the currency of its last dump. Intervening updates are lost. This can result in catastrophic consequences for C^3I systems and is therefore unacceptable. However, Teradata will soon be releasing a software-based rollforward capability (mid to late '85) and plans to announce an Archival Storage Processor (ASP) dedicated to rollback/rollforward operations. With these changes in mind, the DBC/1012 may eventually be considered as good as, or in fact superior to, existing similar facilities provided by other DBMSs.

The Table Rebuild program is a recovery facility which reconstructs corrupted tables. Table rebuild can be accomplished on part of a table, a complete table, or all tables on an AMP. Table rebuild is necessary when the DBC/1012 cannot automatically recover a critical table, or when data is lost due to a disk hardware failure. Tables are rebuilt from backup processors if the FALLBACK option was specified during the initial table definition process.

[illegible] Reconfiguration

[illegible] necessary when the data base is enlarged, when there are changes in the data requirements, when there is a need for faster response time, or when additional host CPUs are installed. To accommodate reorganization or during initial system creation, Teradata provides Configuration and Reconfiguration programs.

The Configuration program describes the processors that compose the system by establishing the configuration map. The processors include IFPs, AMPS and [illegible] that are connected to the system. The Reconfiguration program [illegible] the configuration map to interface these processors into a working unit. When the DBC/1012 is initialized, a configuration map is [illegible] all. When AMPs are added or removed the system can [illegible] reconfigured by using the Reconfiguration program. Data is [illegible] the AMPs and backup FALLBACK clustering [illegible]. The Configuration and Recognition [illegible] may at the DBC/1012 Operator Console. These [illegible] within the realm of a local DBA's responsibility, but [illegible] the System Administrator.

[illegible] times are dependent upon the size of the data [illegible], the amount of data which must be redistributed between AMPs, and the number of AMPs/DSUs involved in the reconfiguration [illegible]. During the 3 Processor Benchmark testing scenario (discussed in [illegible]), the following times were recorded in recofigurating DBC/1012 [illegible] for a 17 table, 51 million row data base:

RECONFIGURATION	TIME
[illegible] to 20 AMPs	15 minutes
[illegible] to 4 AMPs	7.5 minutes

Physical reorganization of the system is not mandatory when columns are added to tables in the data structure. Table alteration can occur independently of system reorganization due to the relational table structure of the DBC/1012. A simple TEQUEL statement makes it possible to add a column to a table. Times to reconfigure tables are dependent on table column position. For example, adding a column at the end of the record does not require reorganization: the system updates the Data Dictionary/Directory and nulls are added to the new column's data entries. If a column in the middle of the table, for example, is dropped, the table reogranization process is very similar to performing an UPDATE-ALL on the table. Times for this operation are comparable to those found in Section 5.3.1.3 for Update operations.

The DBC/1012's reorganization tools are definite enhancements to the data base administration function. Their applicability to C^3I data base environments is especially clear, since they support changing information structures and enable the system to gracefully (and almost transparently) accommodate increased storage requirements and changing access patterns.

6.3.4 Efficiency Tools

Efficiency tools are available to the DBA and users possessing appropriate privileges. The TEQUEL "CREATE VIEW" statement facilitates data manipulation operations. The VIEW statement can be used to combine selected columns from a table or group of tables into a new relationship. The VIEW statement can be used to combine selected columns from a table or group of tables into a new relationship. A VIEW may be used to display parts of a table or to update, insert, and delete table rows of given column combinations [DBC-4:7-10]. Therefore, the VIEW is a powerful tool that can aid during retieval and update by simplifying request specification. VIEWs can be used to rename tables, reorder columns, or join columns of multiple tables.

MACROs can also be used to gain efficiency in data base operation. MACROs are treated as a single "transaction". A MACRO can be invoked from a COBOL program using "BEGIN TRANSACTION.....END TRANSACTION" code, or used from the DBC/1012 TEQUEL environment [DBC-4:9-2]. The purpose of the macro is to create a collection of readily available TEQUEL operations for the Data/System Administrators and users. It is especially useful when routines are executed regularly as in the case of preparing and generating repetitive reports.

The TEQUEL CREATE INDEX statement enables the DBA (or user with the CREATE INDEX privilege) to establish efficient access paths (secondary indexes) to table data, thereby improving data base operation performance. Since secondary indexes may be created after table definition, this facility gives the DBA the ability to "tune" the data base. Secondary indexes may be created or dropped as desired. Index creation can be considered a high performance feature of the DBC/1012. Index creation times observed during the 60-Processor Benchmark were from 31 to 37 minutes for a 5.5 million row table.

There is no provision to name an index. Thus, the user must be aware of all columns which constitute the index during its creation, use, and deletion. Although the user can invoke the "HELP" utility to find out what columns in the table constitute the indexes, we consider the lack of an index names facility an inconvenience.

With the above-noted exception, we consider the efficiency tools offered on the DBC/1012 to be as good as or superior to those found on any high performance DBMS.

6.3.5 Accounting Tools

Several accounting tools are available which provide users and administrators with data base statistics pertinent to their areas of responsibility.

The Data Dictionary/Directory contains detailed information on the tables, views, macros, data bases, indexes, columns, users, accounts, and access rights on the system. To access this information, three distinct classes of views are established; end user, DBA, and System Administrator views. The three levels vary in degree and level of data accessibility.

The end user views present information on each of the data bases, tables, and table columns for which access privileges have been granted. Also, a "UserRights" [DBC-4:12-1] view provides information on the privileges afforded to the "viewing" user.

A DBA can invoke the views described above; additionally, a UserGrantedRights view [DBC-4:12-1] could be accessed to review information on privileges granted to other users in the DBA's data base.

The Data Dictionary/Directory provides the System Administrator with information about system events and resources, such as logons/logoffs, processor utilization, disk space availability, and host queue length statistics. These accounting tools assist the System Administrator in evaluating alternative data base and configuration tuning strategies.

6.3.6 Summary

In our opinion, the DBC/1012 is highly capable of supporting data base administration requirements; the major exception being its current lack of a rollforward utility. Three features are regarded as highly enhancing data base administration operations.

The first feature is the ability to delegate entire data base responsibility and authority through the granting of the CREATE DATA BASE privilege. Large C^3I systems consisting of several data bases could be implemented on a single DBC/1012 configuration under such a delegation strategy.

The second feature is the DBC/1012's ability to restructure data base information and expand the machine configuration to grow gracefully in response to evolving user requirements. No known system provides this degree of sensitivity.

The third feature is that extensive system accounting reports are available on demand to the Data Base Administrator or the DBC/1012 System Administrator. These inlude system utilization reports, user activities and privileges, and data base and table ownership hierarchies.

6.4 Insertion of the DBC/1012 Into An Existing System

Several intelligence data handling systems are currently undergoing upgrades. Some of these, such as the SAC IDHS-80 project, are too far along into the implementation phase to consider using the DBC/1012 as the data management medium. Moreover, once completed, this system, and others like it, will probably not undergo another complete upgrade for at least a decade. In the interim, however, it might become attractive to enhance the system by adding the superior data handling capabilities of newer technology equipment such as the DBC/1012. This section examines the feasibility of inserting the DBC/1012 into existing systems and using it in conjunction with conventional DBMS technology.

6.4.1 Insertion Approaches

The following paragraphs discuss the various levels of insertion of new technology into existing systems, and the methods for accomplishing the modifications. The order of discussion is in ascending order of feasibility and descending order of difficulty. At each level, a different definition of "system" is used. In general, the broader the

definition, the more feasible is insertion. The definitions are not meant to be rigorous according to either data base management science or operations research standards, but, rather, serve the purpose of this discussion only.

6.4.1.1 Tight Systems

We define a tight system as one which exists on a single computer, or which is made up of several computers which are tightly coupled, usually homogeneous, and for which data management services are requested by the applications programs via a procedural language.

Four schemes for inserting the DBC/1012 into such a system have been identified as at least theoretically possible. They are briefly discussed below in ascending order of feasibility, and are presented more for the purpose of highlighting the difficulties involved than to advocate integration approaches.

6.4.1.1.1 Automatic Translation

Software could be developed which would read application source code, identify the procedural calls to the imbedded data management service, and translate them to the equivalent TEQUEL statements. The new application source code would then be re-compiled and would replace the original versions. The original DBMS and disk storage devices would be discarded and replaced with a DBC/1012 and a TDP for the host system.

Even for a well-defined target set, this procedure would be a very large undertaking, at the end of which a complete re-verification of the system would be required. When contemplating the complexities of translating from one data model to another, from procedural to non-procedural DML, and adjusting to contradictory data formats and naming conventions, one begins to see the difficulties inherent in this approach. A final verdict of "infeasible" must be arrived at when it is realized that in such systems,

part of the responsibility for logical data management is usually imbedded in the application program's logic via such practices as remembering (or even modifying!) pointers previously extracted from data records, "re-positioning" files via artifices such as CLOSE and OPEN statements (this practice, furthermore, can be performed at nested levels of file subordination), and in general playing tricks based on the programmer's intimate knowledge of the quirks of a non-rigorous data management service.

6.4.1.1.2 Manual Translation

Manual Translation of all data requests in the application programs is possible only for relatively small systems. The effects of this approach are similar to AUTOMATIC TRANSLATION in that the data management services are replaced and the system must undergo complete re-verification. It is anticipated that some logic changes will also be required for the reasons discussed above. We know of no C^3I systems small enough to be cost-effectively so modified.

6.4.1.1.3 Interpretation

In this approach, the application programs and the existing data management service are left intact. The existing disk storage system is replaced with a DBC/1012, a TDP is added to the host, and an interpreter is implemented to transform the data management service physical I/O requests into appropriate TEQUEL statements. Under this approach, all disk-resident control structures of the original data management service, such as index files, would be defined to the DBC system as data files. The DBC would, in turn, create its own control structures to access this data, thus imposing a redundant level of control on the system, increasing the number of I/Os, the storage requirements, and the processing overhead required to access data. With the exception of the highly problematic possibility of improvements through DBC caching, all effects of this approach tend to degrade system performance.

6.4.1.1.4 Look-Ahead

In this approach, the system is left essentially undisturbed except for additions. The disk storage subsystem is slightly modified to accommodate a microprocessor between the host CPU and the controller(s). The microprocessor contains software which examines the data requests to determine whether the data is on the disk or is presently residing in the DBC/1012. If the latter case is true, a request is sent to the DBC to download a selected sub-set to the disk subsystem. In essence, this is a storage hierarchy such as that implemented in mass storage systems, and might be useful if the system is to be expanded to maintain data collections which are beyond the capacity of the existing disk devices or the capacity of the existing data management service to efficiently manage such large collections. The technical risk of interposing the microprocessor and the potential bottleneck thus introduced are at this time unknown.

6.4.1.2 Loose Systems

A loose system is defined for this discussion as an organization of functional entities devoted to an overall mission. As such, it can be composed of people and procedures as well as computers. The computers can be stand-alone dedicated tools or can be loosely coupled to achieve some co-operation in performance of the mission. The system is considered to be functionally partitioned such that each computer (subsystem) has specific and unique responsibilities. There are two general situations in which new technology might be inserted: replacement of an existing subsystem, or addition of a new subsystem to provide a new capability. In either case, if the subsystem is a stand-alone dedicated tool, the insertion of new technology is neutral technically. If, however, the subsystem is to be interfaced with other computer subsystems, the feasibility of inserting [illegible] of the difficulty of establishing the interface.

6.4.1.2.1 Replacement of an Existing Subsystem

If an existing subsystem is to be replaced with new technology, the previously established interface procedures must be complied with if the change is to be transparent to the rest of the system. If the change is not transparent, the other parts of the system which interface to the subsystem being upgraded will have to be modified to conform to the new interface. If there is no data base transfer activity across this interface, there is no impact involved in the insertion of the DBC/1012. That is, if the DBC/1012 is being inserted for the purpose of supporting only the modified sub-system, its presence causes no integration problems.

6.4.1.2.2 Addition of New Capabilities

If a new capability is being added to the system, and this new capability is essentially independent of the existing data base, its integration costs are not caused by the new technology being inserted. Any interfaces which need to be accomplished, moreover, are NEW interfaces, thus requiring modification of the affected parts of the system, whether or not new technology is being inserted. An example would be the addition to a weapon assignment system of a capability to graphically display the effects of terrain masking on weapons effects. The terrain information is new to the system and is used by no other subsystems. The insertion of the DBC/1012 in this case has as its justification the need to process large subsets of a very large data base within reasonably short time periods, and imposing no additional burden on the existing system. The passing of target(s) location information, retrieved from the central data base, to the new subsystem, must and will be accomplished in exactly the same manner whether a DBC/1012 or conventional DBMS technology is used.

6.4.2 TEQUEL Compatibility with Current DBMS Languages

Insertion of a DBC/1012 into an existing system must also address the compatibility between the new DBMS and the currently employed DBMS. To illustrate the issue of compatibility, comparisons are made between TEQUEL and SQL, and between TEQUEL and MODEL 204.

IBM's SQL was chosen for this comparison because it represents a "standard specification" of the SEQUEL relational data base language. SQL is widely used in commercial installations where a relational DBMS implementation has been specified.

CCA's MODEL 204 is not structurally a relational data base, but does support the construction and utilization of relational views to a data base. MODEL 204 is gaining acceptance especially in the IDHS community, where it has been declared the Department of Defense Intelligence Information System (DoDIIS) standard for intelligence data base management systems.

TEQUEL and SQL have similarities in their Data Manipulation Languages. In fact, it is possible to intermix certain TEQUEL and SQL syntax in data manipulation operations. Therefore, users who are familiar with SQL will be able to write equivalent TEQUEL statements with very little modification to the code. There are, however, some distinct differences in their syntactical and semantical structures which make them substantially different languages, especially in terms of their respective DDL statements. Refer to Appendix E for a detailed discussion of TEQUEL/SQL compatibility issues.

TEQUEL and MODEL 204 are compared in the following pages by illustrating how their respective constructs are employed in several applications examples. Although MODEL/204 supports relational structures, MODEL/204 documentation refers to tables as "files". For the purpose of this presentation MODEL 204 files will contain information equivalent to the DBC/1012 "table".

The examples are based on the following simple files/tables. "DBNAME" is a data base implementation, in which the tables, "CLIENTS" and "PAYMENTS" reside.

CLIENTS
Primary index: ACCOUNT_NO

```
-----------------------------------------------
ACCOUNT_NO  NAME      ADDRESS   TELEPHONE
-----------------------------------------------
```

PAYMENTS
Primary index: ACCOUNT_NO

```
---------------------------------
ACCOUNT_NO  DATE      AMOUNT
---------------------------------
```

Example 1: Print all names and address of account numbers greater than 1000.

```
MODEL 204:  OPEN CLIENTS
            BEGIN
            1. IN CLIENTS FIND ALL RECORDS FOR WHICH
               ACCOUNT_NO IS GREATER THAN 1000
            2. FOR EACH RECORD IN 1
               2.1 PRINT ALL INFORMATION

TEQUEL:     RETRIEVE DBNAME: CLIENTS.ACCOUNT_NO, NAME,ADDRESS, TELEPHONE
            WHERE ACCOUNT_NO > 1000;
```

Example 2: Print all of John Doe's payment dates and amounts

```
MODEL 204:  OPEN CLIENTS
            OPEN PAYMENTS
            BEGIN
            1. PRINT "JOHN DOE"
            2. IN CLIENTS FIND RECORDS FOR WHICH
               NAME = 'JOHN DOE'
            3. NOTE ACCOUNT_NO
            4. IN PAYMENTS FIND ALL RECORDS FOR WHICH
               ACCOUNT_NO = VALUE IN 3
               4.1 PRINT DATE AND AMOUNT
```

Alternately, MODEL 204 permits many files to be combined into one physical file containing multiple logical record types of varying lengths. The records must have one field in common:

```
            1. PRINT "JOHN DOE"
            2. FIND RECORDS FOR WHICH NAME = 'JOHN DOE'
            3. NOTE ACCOUNT_NO
            4. FIND ALL RECORDS FOR WHICH
               ACCOUNT_NO = VALUE IN 3
               4.1 FOR EACH RECORD IN 4
               4.1.1 PRINT DATE AND AMOUNT
```

```
TEQUEL:     RETRIEVE DBNAME: CLIENTS.NAME, PAYMENTS.DATE, AMOUNT
            WHERE CLIENTS.ACCOUNT_NO = PAYMENTS.ACCOUNT_NO;
```

This example implements a "join" between the tables by using the "WHERE" clause.

Example 3: Print a sorted list (by account number) of all "Smiths" who are older than 18.

```
MODEL 204:  OPEN CLIENTS
            1. FIND ALL RECORDS FOR WHICH
               NAME = SMITH
               AGE IS GREATER THAN 18
            2. FOR EACH RECORD IN 1
            3. PLACE RECORDS IN 2 ON THE LIST GENERAL
            4. SORT LIST GENERAL BY ACCOUNT_NO
            5. FOR EACH RECORD IN 4
               PRINT NAME AND ADDRESS

TEQUEL:     RETRIEVE DBNAME: CLIENTS.NAME, ADDRESS
            WHERE AGE > 18
            ORDER BY NAME, ADDRESS
            WHERE NAME CONTAINS "SMITH"
```

This example shows how MODEL 204 and TEQUEL both have pattern matching capabilities. Note that in this example, "SMITH" may be a first name, last name, or part of a name; e.g. "SMITH-JONES". Field definitions, such as LNAME could prevent this ambiguity.

Example 4: This last example prints a list of customers and the number of orders outstanding. It assumes two tables/files: VEHICLES and CLIENTS.

```
MODEL 204:  OPEN VEHICLES
            OPEN CLIENTS
            BEGIN
            1. IN CLIENTS FIND ALL RECORDS FOR WHICH
               POLICY_HOLDER = "COLLEEN JOHNSON"
```

```
2. FOR EACH RECORD IN 1
   2.1 NOTE POLICY_NUMBER
   2.2 IN VEHICLES FIND ALL RECORDS FOR WHICH
        POLICY# = VALUE IN 2.1
   2.3 COUNT RECORDS IN 2.2
PRINT POLICY_NUMBER AND COUNT IN 2.3
END
```

TEQUEL: (There are two implementations provided.)

```
RETRIEVE VEHICLES.POLICY# COUNT VEHICLES.POLICY#
WHERE POLICY# IN
      (RETRIEVE CLIENT.POLICY_NUMBER, POLICY_HOLDER
      (WHERE POLICY_HOLDER = 'COLLEEN JOHNSON');

RETRIEVE VEHICLE.POLICY# COUNT VEHICLE.POLICY#
WHERE CLIENT.POLICY_HOLDER CONTAINS 'COLLEEN JOHNSON'
AND CLIENT.POLICY_NUMBER = VEHICLE.POLICY#;
```

The first TEQUEL request is similar to the MODEL 204 technique except that the subquery is inverted. The second TEQUEL request demonstrates how tables can be joined on common data between columns.

Two major differences exist of TEQUEL and MODEL 204. The first is that TEQUEL is non-procedural, and MODEL 204 is semi-procedural. The second is in the file structure and handling methods employed by the respective DBMSs. For the examples noted above, TEQUEL seemed to require less coding to achieve the same results. The language constructs were quite different in some cases.

The MODEL 204 Data Base layout is very flexible in the sense that multiple logical files can be maintained in the same physical file. This simulates the concept of relational tables that TEQUEL uses. Additionally, MODEL 204 is designed to work on a Network or Hierarchical structure. The only drawback is the overhead in maintaining the indexes and System Tables (A,B,C,D) required by the system to operate properly.

6.4.3 Summary

Insertion of the DBC/1012 into an existing system environment has a varying range of effects on an organization. The best possible circumstance is where an organization employs a relational architecture for its data base assets and is introducing new capabilities which will interface minimally to other subsystems in the performance of an overall mission. In this case, the host/DBC-1012 configuration is, in essence, separate from existing subsystems and bound to the overall system only by external data interfaces. The worst possible circumstance is where the installation of a DBC/1012 forces a complete reorganization of the data base from both a physical and a perspective view (say, from a network architecture to the required relational-based DBC/1012 architecture). Besides performing the data base conversion process, all existing applications programs or interactive procedures must be cast aside. This is, in fact, a re-implementation, not an upgrade via insertion. The additional problem of maintaining an operational data base while configuring the new design introduces several problems well-known to those experienced in the field, such as synchronization, cutover delay, and the costs associated with the extra labor required to maintain two data bases at once.

SECTION 7

ASSESSMENT SUMMARY AND RECOMMENDATION

This section presents observations and responses to questions raised in our Technical Proposal and issues surfaced during the effort. Recommendations on the applicability of the DBC/1012 to the C^3I data management domain are given in the remaining pages of this section.

7.1 General Observations

The following are summary findings. Where applicable, the reader will be directed to other sections of this report or previous project reports.

7.1.1 Indexing

For the current release of the system, the searching algorithm is depicted in Figure 6-1 of the Capabilities and Characteristics Report [MCC-1], and modified to show B-Tree search. Some pertinent points of information supplement this figure:

- For equality searches, only one AMP is involved in a primary key retrieval. This gives a linear throughput curve (transactions vs. AMPs).

- All secondary searches, and primary searches on ranges, are broadcast to all AMPs, thus, addition of more AMPs will not give a corresponding increase in throughput capacity.

- The search (B-Tree) gives a logM of N DSU access rate (see Section 5), thus making possible the searching of files containing a very large number of records in relatively few DSU accesses.

Teradata states that, for the next release, a hashing capability for secondary indexes will be instituted which will identify the AMP which is responsible for that record, thus involving only one AMP in a secondary retrieval, giving a linear throughput curve similar to that for primary searches. Of course, this improvement will only apply to equality searches.

7.1.2 The Effects of Data Base Imbalance on Parallelism

In the DBC/1012, data is stored according to a hash value computed using the defined primary index. The concern is that, if the result of the calculations locates data on the DSUs in a distribution which tends to be "clumpy", the benefits of parallelism may be lost.

7.1.2.1 Possibility of Imbalance

There is a distinct possibility of data base imbalance in C^3I applications if care is not taken in the selection of the primary index of a table. The danger of imbalance is greater with a non-unique primary index than with a unique primary index. In C^3I applications, there are many possible types of keys which lend themselves to being primary indexes, from the applications aspect, but which are not unique and which tend to be clumpy. Moreover, the unique keys which would be best as primary indexes are not usually useful as search keys. For instance, a message file contains the following potential keys:

- Message ID
- Source ID
- Country Code
- Equipment Type
- Date-Time Group

Of these fields, only the Message ID is unique, and it is not very useful to search on. Thus, the most efficient search argument is mostly unusable. All of the other fields have potential clumping problems. Source ID will tend to clump for those sources which are most active, and there is a wide range of relative activity among the various sources. Country Code will clump, roughly, according to the size and military or diplomatic characteristics of the country. Equipment type will clump according to the number of sightings made per type. There will be a great difference in the number of messages relating to trucks than to aircraft carriers. This may be a bad example in that few C^3I systems will be concerned with both of these equipments, but the general idea is there. Date-Time Group will tend to clump according to periods of relative high activity such as scheduled production shifts and periods of crisis. This clumping is ameliorated due to the granularity of the "time" (minutes), but is exacerbated by the tendency of automated C^3I systems which gather or produce "messages" (actually reports) over the extent of a production shift and then disseminate them as a large batch during a very short period of time.

One type of C^3I system which perhaps lends itself to an efficient primary key use is that of reconnaissance exploitation. A considerable amount of the work is carried out by installation. The BE Number used to identify installations appears to exhibit the properties recommended for a primary key. Although it is not purely unique, it is almost unique, and the values exhibit a fairly even distribution and are fairly compact within the clumps. This means that the BE Numbers which are clumped around strategic locations will tend to be assigned to different AMPs ("dense and contiguous", from System Manual [DBC-3:2-3]).

7.1.2.2 Effects of Imbalance

Imbalance in data residency can, for the general case, have two degrading effects on effective parallelism and system performance. First, imbalance can degrade the response time to a single user if that user's data is imbalanced. Quite simply put, during a significant period of time during the processing of the request, only one component is working on it. Second, if the system must finish one request before it can process another, the entire system is idle while waiting for one component to finish its work.

The latter effect is the more serious of the two. The degree to which it is important in the DBC/1012, however, depends on the mix of requests. In the first place, the effect is lessened for those requests which require sorting: an ORDERED retrieval or a JOIN. If no sorting is required, the AMPs which finish early can pass along their results and turn to the next request. This benefit, however, is only fully realized if the other requests are single AMP operations, such as a single row retrieval. If they require all AMPs, eventually the burdened AMP will affect the others. Even in this situation, however, the effect can be damped by judicious assignment of request priorities so that degrading requests can be side-tracked until such time as the system can deal with them.

The effective speed of the DBC/1012 is such that it is difficult to imagine a group of interactive analysts large enough to cause a significant burden on it. The major concern is the support of production software which has heavy needs for data.

We anticipate that there will be no significant degradation of parallelism for applications such as target and weapon assignment, even though the large amounts of data they require tend to be conceptually

(geographically) clumped. This data seems to satisfy the "dense and contiguous" recommendation, and the types of retrievals are heavily weighted toward single row responses (Target Id, Weapon Type Cnaracteristics, etc.). The assignment algorithms, however, quite possibly require ORDERED data (e.g., terrain elevations ORDERed by lat/lon, so the matter deserves a more exhaustive analysis.

Tne inrerencing work done in support of I&W analysis will probably require a heavy proportion of JOINs to dynamically establish relationships between previously unrelated information. Again, this is not seen as a problem for the support of I&W analysts, since they cannot enter requests fast enough to burden the machine. This point could be argued, certainly, from the standpoint that a single monstrously complex request, requiring numerous tables and all data rows, could burden the machine. However, in a multi-user environment, degradation would be limited due to safeguards included in the DBC/1012 operating system software. The operating system maintains a process priority scheme similar to those employed in other modern operating systems. Processes are assigned priorities based on their complexity, resources required, and age. The approach in achieving acceptable throughput is quite similar to any multi-user system. Future software releases will allow classes of priority value inputs to be set up prior to operation; now all operations are treated in the same way by the system.

It is in the area of production AI software, applied, perhaps, to the requirements of autonomous battle management or strategic missile defense, that the problem may become too much for a DBC/1012. By the time that this application becomes a reality, the DBC/1012 will have become obsolete, anyway.

7.1.3 Bottleneck Possibilities at the Ynet

One of our original major concerns, expressed in our proposal, was that the Ynet was the only component of the DBC/1012 which could not be replicated as other components were. That is, as more and more IFPs and AMPs were added to the system to meet increased loading requirements, the number of Ynets remained at two. A potential bottleneck was suspected. The truth of the matter is, of course, that the Ynet is, in fact, being replicated as the system grows. It is a parallel pipelined tree structure, with additional nodes being added as the number of processors increases. Because of this parallel pipelined mode of operation, the crucial governing statistic is that of simple throughput capacity - the number of bytes delivered per second. At a byte rate of 6MB per second for one of the duplicated Ynets, bytes can be delivered to 1024 processors at the rate of 5859 bytes per second to each processor. These bytes, moreover, are for the most part "finished" information, chaff having been discarded at the AMP level before reaching the Ynet.

Although JOIN operations can impose a significant burden on the Ynet due to AMP-to-AMP communication, they impose a greater burden on the AMPs, thus, for JOINs examined, the Ynet was not a bottleneck.

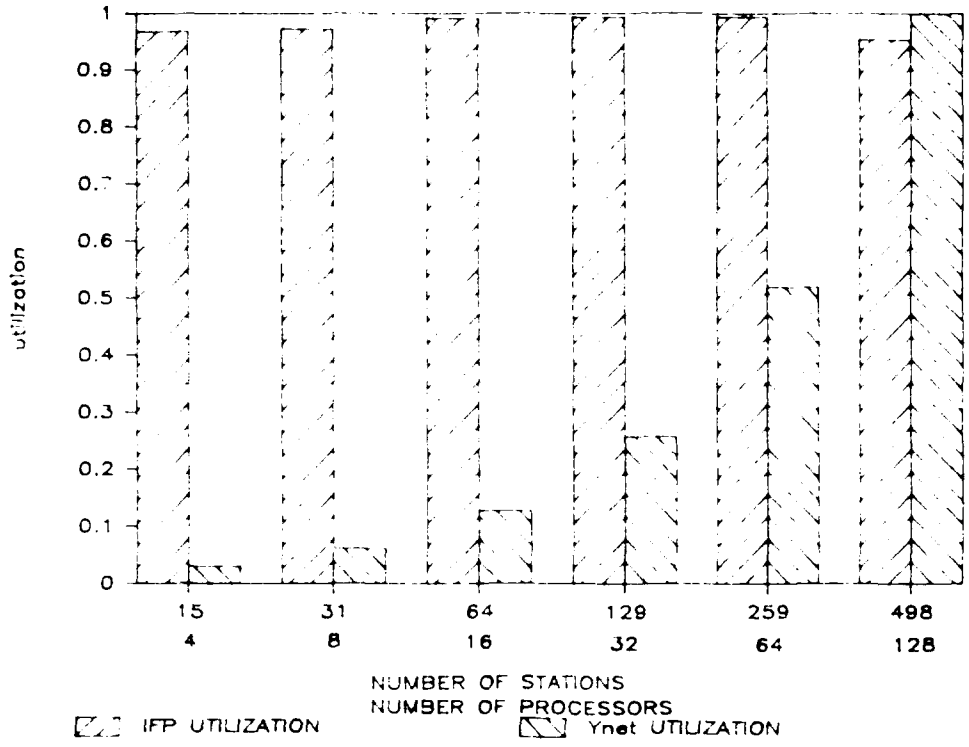
It was decided that the type of operation most likely to cause a bottleneck at the Ynet would be one which caused great amounts of data to be delivered to the end user while at the same time involving relatively little participation by the AMPs and IFPs. The operation we chose to meet these criteria was a non-interactive RETRIEVE on a unique primary key. Such an operation will involve only those AMPs which can satisfy the request (i.e., the request is not broadcast to all AMPs). The non-interactive characteristic is felt to be that type of operation which has the highest proportion of similar requests which can be cached to avoid parsing, thus lowering the activity in the IFPs. Finally, the RETRIEVE should return a lengthy answer for each request, thus causing a large data flow back to the user over the Ynet.

A hypothetical C^3I application was posited, which is roughly termed SITUATION DISPLAY. It has the following characteristics:

- A graphical display of the "situation" which includes a dynamic background is presented to analysts at their individual workstations.
- The workstations are 1000 x 1000 pixel CRTs.
- A pixel is defined in the data base by an eight-bit byte.
- The full screen is automatically updated once each minute for each analyst.
- The rows of the data base are 30,000 bytes in length (the DBC/1012 maximum), which means that to update a full screen, 33.33 rows must be RETRIEVED.
- To add a semblance of reality, an UPDATE operation is performed in which one row is updated each second. This activity is constant, independent of the number of stations being supported. It is considered to be arriving at the installation from external sensor systems and is not being operated by the analysts.

Figure 7-1 (upper and lower) shows the results of this experiment. The Ynet has become the bottleneck, determining the upper limit of support at 498 stations. At this level of support, 128 processors (64 IFPs and 64 AMPs) are configured.

While this experiment did show a potential saturation condition for the Ynet, it should be noted that a rather severe requirement was modeled. In no case, when modeling the more reasonable types of operations, such as PI



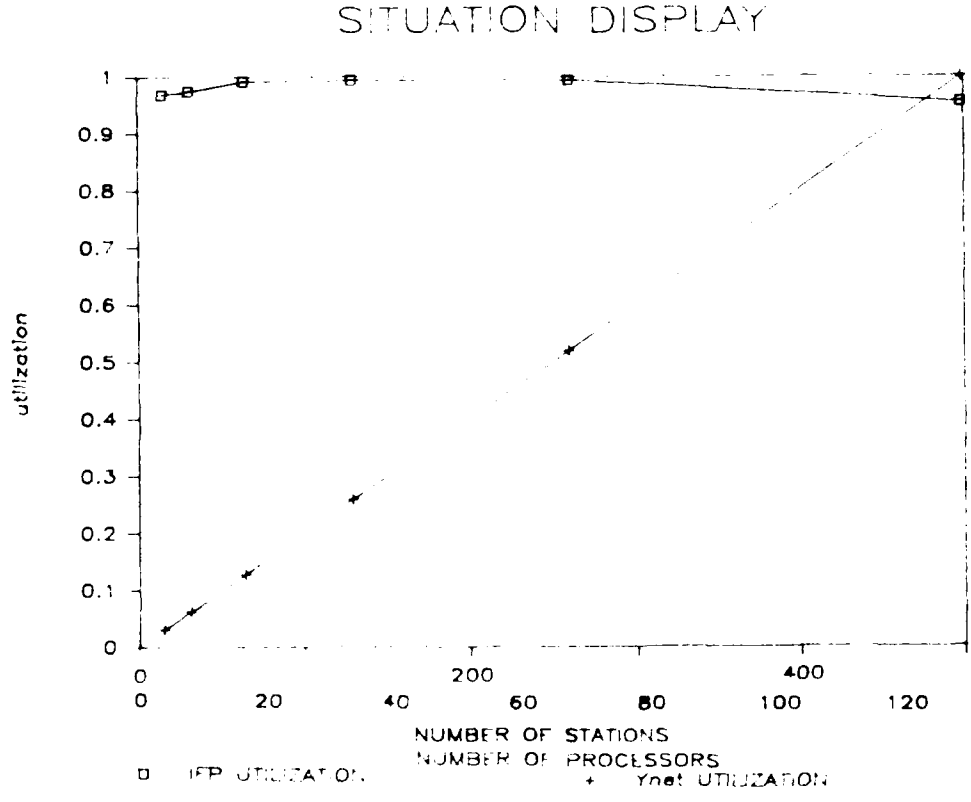


Figure 3.1 Sample Situation Display Model outputs

and I&W interactive sessions and generic operations involving data access characteristics more in the range of most applications, did the Ynet exceed 8% utilization, even for configurations of 1024 processors.

7.1.4 Miscellaneous Observations

7.1.4.1 JOIN Operation Efficiency

Achieving efficient JOIN operations in the DBC/1012 was a complex design task because of the distributed data base architecture and parallel processing nature of the machine. On other processors, table rows are usually found on the same disk volume, whereas on the DBC/1012, single tables are distributed across many different DSUs all controlled by their respective AMPs. Thus a JOIN may necessitate creation of spool files, or transfer/replication of data to one or all AMPs, in order to construct JOIN tables before the JOINs can be executed.

The TEQUEL Optimizer examines JOIN source statement requests to determine the most efficient JOIN strategy. Information obtained from the Data Dictionary/Directory, such as key types and composition, numbers of table rows, and uniqueness of keys is used.

The two major components in "JOIN Planning" are the JOIN geography and the JOIN step.

JOIN geography ("geog" in Teradata terms) refers to the preparation necessary to accomplish a specified JOIN step. Four types of geog exist:

- Dup - Duplication of rows
- Hash - Rows are hashed, possibly sorted, then redistributed to appropriate AMPs

- Local - Rows are already on the right AMP; may require a sort step

- Direct - Equality relation exists between matching keys on the table. Most likely will lead to the best JOIN plan.

These four geogs, when applied to either the left or the right side of a JOIN result in 16 different (but not necessarily logical) combinations for preparing the left and right tables for the JOIN.

The JOIN step, as it is referred to in Teradata technical documentation, is the process of creating an output from the original tables or spool files. Four forms of the JOIN step are:

- Product JOIN - Each and every row of the left table are compared with each and every row of the right table.

- Exclusion Product JOIN - Same as Product JOIN except that result rows are created for each row of the left table for which no row in the right table passes the JOIN constraint.

- Merge JOIN - Source tables are in row hash order. Only row combinations with matching hash codes are compared.

- Exclusion Merge JOIN - Same as Merge JOIN except that results are created when each hash match does not pass constraints.

JOIN cost estimates are based upon:

- Cost of preparing tables for the JOIN step
- Cost of the JOIN step
- Cardinality (number of rows estimated for prepared JOIN inputs)

The contribution of each cost component varies with the general type (Merge, Exclusion) of JOIN step.

Index types, such as primary or secondary, of the specified tables also a factor in determining cost.

The preceding text is a general overview of the extensive work that Teradata has conducted in attempting to optimize JOIN processing efficiency. The JOIN strategy can also be reviewed by the user prior to JOIN execution by invoking the EXPLAIN command. This allows the user to inspect the effects of a JOIN operation, optimize the request (if desired), and review the predicted results of the new JOIN request. In summary, Teradata not only provides an optimization-driven approach to performing JOINs, but also provides the user with tools to enable the development of optimization-driven requests.

7.1.4.2 Effects of the Parsing Loads on IFPs

Because of the heavy loads associated with parsing, the IFP becomes a bottleneck for operations which are inexpensive in terms of data access, such as unique primary key retrievals. Model outputs in Appendix D demonstrate this fact. However, parsing loads are mostly associated with ITEQ sessions and a series of dissimilar requests under BTEQ and under program-generated requests. Caching will occur under BTEQ and application programs - driven sessions for similar requests, where techniques such as parameter passing can be employed to reduce parsing loads.

If unanticipated heavy interactive usage is found to degrade system performance, the solution is simple and very effective: buy more IFPs.

7.1.4.3 Location of the Data Dictionary/Directory

The Data Dictionary/Directory is located on each and every DSU (DSU - system disk on 1 ISU/1AMP arrangements) in the configuration. Data dictionary tables for every data base are stored at each DSU.

7.2 Recommendations

Mc^2 recommends that the DBC/1012 should be given serious consideration in C^3I upgrade efforts, and especially for the development of new C^3I subsystems. Two fundamental reasons for making this recommendation are:

- Being a data base machine, it has inherent advantages over general purpose host/DBMS configurations; and
- Its unique performance and reliability characteristics.

The following text explains these reasons in detail. Paragraphs refering to data base machine attribute-related advantages are labeled [DBM] and paragraphs which underscore the DBC/1012's capabilities are labeled [DBC/1012].

7.2.1 Expedites Upgrades [DBM]

Stressing the data base as a "node" in the C^3I architecture and supporting it with a dedicated DBM results in the development of an enduring resource. When processor upgrades become necessary for host and support systems, costs are minimized due to the following reasons:

- No new DBMS is required for the host
- No DBMS applications design and coding costs are expended, except for new host/DBM interface

- Time and money costs associated with initial data base loading operations on the new systems are eliminated (which also eases the burden on the C^3I mission related production requirements). These costs are directly proportional to the size of the data base.

7.2.2 Configuration Planning is Straightforward [DBC/1012]

The key feature of the DBC/1012's architecture is the isolation of functionality. Because of this isolation, the architecture is easily modeled. Thus, performance assessment tools can be (and have been) built to forecast the support that the DBC/1012 can provide an application. Parameters, such as data base size, user transaction traffic, and system operational requirements (for example, required response times) can be used to determine the optimum DBC/1012 configuration.

Additionally, DBC/1012 configurations are unconstrained by technical or merchandising policies, except for the minimum 2-IFP/4-AMP system. Flexibility in configuration component types and quantities supports an economic Buy-What-You-Need policy. The Need is clearly established by analytic modeling of the customer application.

7.2.3 Facilitates System Expansion [DBC/1012]

As the user data base grows, The DBC/1012 configuration can grow in line with the expansion of the data base. Additionally, requirements mandating an increase in aspects of the data base processing performance (faster responses, more users, more complex query operations, etc.) can be modeled and appropriate components (IFPs, AMPs, DSUs) added to the system. Reconfiguration executes under software utility control and can be performed concurrently with normal data base production operations.

7.2.4 Non-Stop Running [DBC/1012]

The DBC/1012 exhibits high reliability due to its redundant architecture. Under a fallback policy, there is little or no apparent degradation when one component in a cluster of AMPs/DSUs fails. Reconfiguration to the backup AMP/DSU takes place automatically. Few errors are system wide (refer to Section 4.5 of this report). Also, repairs can be effected on the down equipment while the rest of the equipment is on-line. Once the component is repaired and on-line, current data base information is reloaded automatically from the fallback processor.

7.2.5 Multiple Host Operation [DBC/1012]

The DBC/1012 has the ability to support data base operations under a multiple host environment. In the C3I processing domain, this is seen as an advantage since various organizations can exploit the data bases through equipment most suited to their application, and at the same time, benefit by an integrated scheme which supports center-wide data transfer and utilization. The most serious shortcoming under this approach is the limited availability of host interfaces currently available for the DBC/1012. Teradata has stated that efforts to expand interfacing capabilities beyond the IBM and Plug Compatible Machines are ongoing; however when these interfaces will be available is currently unknown.

7.2.6 Superior Performance and Capacity [DBC/1012]

The benchmark results and the model outputs demonstrate that the DBC/1012 can and does support data base transactions of a rate and complexity that only extremely large host/DBMS configurations can achieve in limited cases. In most cases, there is no other machine that can approach these performance characteristics.

Also significant is that DBC/1012 expansion introduces no control burden overhead and minimal data traffic overhead (if any) due to increased status messages over the Ynet. (This traffic is minimized due to the message priority scheme used to handle Ynet message traffic).

The terabyte data capacity of the DBC/1012 can be matched by traditional high-end mainframes but at an enormous peripheral processing burden expense. More on this subject will be presented in the next subsection.

7.2.7 Cost/Performances Advantages Over Traditional Architectures

The DBC/1012 is not always a low cost alternative to general system DBMS configurations from the system acquisition cost perspective. However, it is economically superior to other systems when measured on a cost per MIP basis; i.e., performance potential per unit cost. Obviously, DBC/1012 MIPs are not available for general purpose processing; however, they do represent a measure of its high performance potential for data base processing activities.

The following discussion uses approximate cost data extrapolated from vendor literature. No explicit price quotes were obtained, but the subject information is presumed to be close enough to support this comparison.

The base list price of a DBC/1012 2-IFP/4-AMP system is approximately $211,000 for the hardware and $120,000 for the full set of software (precompilers, CICS, etc.) as of April, 1985. For $331,000, a 2.4 MIP (.6 MIPs per processor) machine consisting of a DBMS, six processors accompanied by twelve megabytes of memory, and two gigabytes of disk storage is provided. This price is somewhat higher than GSA costs for an IBM 3081 K with two gigabyte disk (about $60,000). The IBM 3880 Storage Director is used in this example to support the disk. Low end 3880 Model [illegible] approximately $43,000 (GSA). A commercial DBMS ranges in

price from $100,000 to $200,000. The total price for the storage control, disk subsystem, and DBMS packages thus ranges from approximately $209,000 and $309,000. Processor cost is not included in this price.

Cost comparison at the high end of the configuration spectrum also yields interesting results. Using the Teradata Configurator, an approximate cost of $38.5 million was calculated for a 64-IFP/960-AMP, 1024-processor system. This estimate is subject to several qualifications. First, the Configurator Program could not price the entire system; it priced a system half as large and the resultant cost was doubled (since no Teradata system approaching this size has ever been ordered.) Second, commercial schedules were used (Teradata is currently awaiting GSA schedule approval as of this writing). Third, Teradata suggests that substantial cost savings may result from negotiations on a purchase of a system of this magnitude. This system includes two DSUs per AMP and provides a total storage capacity of 988.80 gigabytes.

This 409.6 MIP machine-driven system can support prime key retrieval operations at a rate of almost 450 prime key requests per second under optimum batch (BTEQ) mode of operation. (This number is extrapolated from benchmark tests. Analytic model execution results approaching 5000 transactions per second have been obtained from applications-driven optimal (balanced) configurations. See Appendix D for outputs).

The IBM 3380 AD4 disk storage subsystem is again used for comparison, since it has the potential to be upgraded to a four gigabyte device and is currently considered state-of-the-art technology. The equivalent data storage capacity requires the support of 247 upgraded 3380s, resulting in an approximate cost of $22 million. These disks must be augmented by 3880 Storage Controllers, one on each host channel. Assuming a high-end IBM 3084Q processor host with 48 available channels, 40 of which would be available for 3880s, results in a cost of approximately $2 million for storage control devices (using the low-end 3880 Model 003 controllers).

Thus total price for the IBM disk subsystems and control units is approximately $24 million. No mainframe price is included, but it is assumed that a 3084 processor (rated at approximately 29 MIPs) would cost approximately $5 million to $6 million. Data processing problems may be severe under this $29 - $30 million system because of the 6:1 ratio between 3380s and 3880 storage control subsystems, since the drives are daisy-chained on the storage control subsystem. Data accessing routines would perform searches over control/indexing information for 12 gigabytes of information on the average.

Data access performance improvements may be obtained by using the 3880 Model 23 Storage Control Application Data Subsystem, which includes cache area. These units range in price (according to the IBM GSA FY85 Schedule) from $105,000 to $192,000. Substituting the IBM 3880 Model 23 in place of the Model 003s in the overall configuration yields a total cost ranges of from $32.2 million to $35.8 million. No estimates are available on the peak transaction rates which could be achieved by this system.

Other, more definitive, price/performance data has been published in Teradata benchmark results. In the 60-Processor Benchmark, discussed in Section 4 of this report, price/performance ratios were from five to six times more efficient than benchmark competitors.

7.2.8 Summary

Regardless of the philosophy adopted by the reader concerning the benefits of data base machines in C^3I environments, the DBC/1012 deserves serious consideration as a system component. It has been shown to have superior capacity and performance over conventional systems. It is highly reliable and fault tolerant. It can interface to multiple host configurations, thereby supporting its economical use and efficient exploitation of data base information. Perhaps, most importantly, the DBC/1012 can be configured to support stated systems operational requirements as they currently exist and support graceful system growth as operational requirements evolve in time.

APPENDIX A

REFERENCE LIST

AFA-1 Winkler, A., et al; The Data Administrator's Handbook; USAFA-TR-76-1; United States Air Force Academy; NTIS; 1976.

BG-1 Bernstein, P.A., and Goodman, N.; "Concurrency Control in Database Systems"; ACM Computing Surveys; Volume 1, Number 2; ACM, June 1981.

CCA-1 Model 204 DBMS User Language Manual; Computer Corporation of America; September, 1984.

CSC-1 Trusted Computer System Evaluation Criteria; CSC-STD-001-83, Department of Defense Computer Security Center, Fort George G. Meade, Maryland; August 15, 1983.

DBC-1 DBC/1012 Data Base Computer Concepts and Facilities; Release 1.0, C02-0001-00; Teradata Corporation, April 1, 1983.

DBC-2 DBC/1012 Data Base Computer Concepts and Facilities; Release 1.1, C02-0001-00; Teradata Corporation, April 1, 1983.

DBC-3 DBC/1012 Data Base Computer System Manual; Release 1.0; C10-0001-00; Teradata Corporation, August 1, 1983.

DBC-4 DBC/1012 Data Base Computer User's Guide; Release 1.0; C09-0001-00; Teradata Corporation, August 1, 1983.

DBC-5 DBC/1012 Data Base Computer Planning Guide; Release 1.0; C07-0001-00; Teradata Corporation, November 1, 1983.

DBC-6 DBC/1012 Data Base Computer Host Interface Manual; Release 1.0; C12-0001-00; Teradata Corporation, September 15, 1983.

DBC-7 DBC/1012 Data Base Computer Operations and Utilities; Release 1.0; C11-0001-00; Teradata Corporation, March 1, 1983.

DBC-8 DBC/1012 Data Base Computer Operator's Guide; (Preliminary); Release 1.2, C15-0001-00; Teradata Corporation, January 1985.

DBC-9 DBC/1012 Data Base Computer CICS Interface Manual; Release 1.0; C14-0001-00; Teradata Corporation, December 3, 1983.

DBC-10 DBC/1012 Summary Information; Teradata Corporation; no date.

DBC-11 DBC/1012 Data Base Computer Reference Manual; Release 1.3; C03-0001-01; Teradata Corporation; March 1985.

DBC-12 Technical Materials from April 8 - 12, 1985 Data Collection Trip.

FLO-1 Flores, Ivan; Computer Sorting; Prentice-Hall; Englewood Cliffs, NJ; 1969.

HAR-1 Secure DBMS; RADC-TR-81-394; Harris Corporation; February 1982.

HT-1 Cook, R.; "Conquering Computer Clutter;" High Technology; Vol. 4, No. 12; pp. 60-70; December 1984.

KOH-1 Kohler, Walter H.; "Survey of Techniques for Synchronization and Recovery"; ACM Computing Surveys; Volume 1, Number 2; ACM; June, 1981.

MAR-1 Martin, J.; Computer Data Base Organization; Second Edition; Prentice-Hall; Englewood Cliffs, NJ; 1977.

MB-1 Notes from 1984 Minnowbrook Workshop on Data Base Machines. Ken Cairns of Teradata.

MCC-1 Teradata DBC/1012 Data Base Machine Capabilities and Characteristics Report (DRAFT); Measurement Concept Corporation; March 1985.

MCC-2 Analysis of the Teradata DBC/1012 Data Base Machine's Applicability to C^3I Operations; Measurement Concept Corporation; July 1985.

NEC-1 Neches, P., et al; United States Patent Number 4,412,285; Teradata Corporation; Filed April 1, 1981; Granted October 25, 1983.

NEC-2 Neches, P.; United States Patent Number 4,445,171; Teradata Corporation; Filed April 1, 1981; Granted April 24, 1984.

RUS-1 Rushby; J.M.; "Specification and Design of Secure Systems;" Pre-Print of Paper to appear in a Pergamon Infotech State of the Art Report on 'System Design;' March 31, 1981.

TF-1 Teorey, T.J., and Fry, J.P.; Design of Database Structures; Prentice-Hall; Englewood Cliffs, NJ; 1982.

TM-1 "Technical Memorandum #1"; Measurement Concept Corporation; March 6, 1985.

ULL-1 Ullman, U.D.; Principles of Database Systems; Second Edition Computer Science Press, Inc.; Rockville, MD 1982.

APPENDIX B
TERMS AND ABBREVIATIONS

In the interest of maintaining accuracy in term definitions and abbreviations, the following terms and abbreviations have been extracted from the Teradata DBC/1012 Data Base Computer Reference Manual. Variations and/or updates were indicated by parenthesis.

Access Module Processor (in hardware) - A processor module in a DBC/1012 system that is primarily responsible for storing and manipulating information to satisfy requests by users. Abbreviated AMP.

Administrator (in Data Dictionary/Directory) - A special user responsible for allocation of the DBC/1012 to a community of users.

Aggregate Operator (in TEQUEL) - An operator such as COUNT, SUM, AVERAGE, etc., that produces a single result from a number of rows.

AMP (in hardware) - Access Module Processor.

AMP Board (in hardware) - A PWBA that provides the processor and controller sections of an AMP module.

AMP Cluster (in hardware) - A group of AMP modules (typically in different cabinets in a large system) that provides fallback capability for each other. A large system typically consists of many clusters of 2 to 16 AMP modules each.

AMP Module (in hardwre) - A complete Access Module Processor, consisting of an AMP processor/controller board, a memory board, two Ynet interface boards, a cable adapter board, and DC power supply.

Application Program (in host software) - A program that performs a particular function or set of functions that the user desires to perform, as distinct from the MVS operating system or the Teradata Director Program. Application programs may be supplied by Teradata or written by the user. Teradata-supplied application programs include ITEQ, the COBOL Preprocessor, Bulk Load, Framer, and the Host Utility. Users can write application programs for the DBC/1012 using either the COBOL or PL/1 Preprocessors, or the Call-Level Interface.

Argument (in TEQUEL) - A value, coded in the invocation of a macro or other TEQUEL statement, that is substituted for the corresponding parameter when the macro or statement is executed.

Attribute (in TEQUEL) - A property shared by all of the fields in a column. TEQUEL attributes include data type, FORMAT, default value, column title, etc.

Backout (process) - The process by which changes to a data base are reversed after a transaction has been aborted, so that the data base is restored to its state as of the beginning of the transaction.

Batch (in host software) - Application programs that run in a background mode, where their execution is not under the direct, moment-to-moment control of a human user.

Block (generic term) - A set of records, rows, or packets that is manipulated as a unit, typically for efficiency of execution. In the DBC/1012, a block is the unit of transmission on the Ynet, channel, and between an AMP and a DSU.

Block Multiplexer (in hardware) - One of the kinds of IBM channels. The other kinds are byte multiplexer, selector (obsolete), and data streaming. The IFP supports only the block multiplexer channel.

BTEQ (software component) - Batch TEQUEL. A host-resident application program, supplied by Teradata as an extra-cost option, that enables a user to submit one or more TEQUEL statements in batch mode for DBC/1012 processing.

Bulk Data Load (software component) - A Teradata-supplied application program that rapidly loads data from files (or programs) on the host computer into tables resident on the DBC/1012.

Byte (in TEQUEL) - A data type in which information is stored as a sequence of zero or more 8-bit elements without translation. Also, one such 8-bit element.

Call-Level Interface (CLI) (software component) - A set of entry points provided by Teradata to facilitate low-level communication between application programs running on the host and the DBC/1012.

Channel (in hardware) - The means by which an IBM central processor is attached to peripheral units. The path by which data is transferred between the host and the DBC/1012.

Channel Interface Controller (in hardware) - The logic (hardware and firmware) in an IFP that manages the DBC/1012 side of the host-to-DBC/1012 interconnection via block multiplexer channel. In terms of IBM channel protocols, the Teradata Channel Interface Controller emulates a channel control unit with two devices attached.

Character (in TEQUEL) - A data type in which information is stored as a sequence of zero or more 8-bit elements, translated into an internal representation on the DBC/1012. Also, one such 8-bit element. Cf: byte.

Character String (in TEQUEL) - A column or constant of character data type.

CICS (in host software) - Customer Information Control System. An IBM program product that acts as a supervisory or "monitor" program for application programs that are optimized for real time interaction with users to perform relatively constrained information processing tasks. CICS runs under control of the MVS operating system (there is also a DOS version of CICS) and communicates with a network of terminals. Application programs written for CICS must use only CICS system services and must obey a number of other constraints imposed by the CICS environment.

Class (in hardware) - A set of processor modules in the DBC/1012 to which Ynet message can be addressed.

Cluster (in hardware) - AMP cluster.

COBOL Preprocessor (software component) - A program that facilitates productive design, coding, and testing of user application programs written in COBOL by processing TEQUEL statements that are embedded within the COBOL programs.

Collision (in hardware) - When two or more processor modules attempt to send packets on the Ynet at the same time, a collision occures. The Ynet will accept one of the packets, and direct other processors to send their packets again at a later time.

Column (in TEQUEL) - In the relational model, data bases consist of one or more tables. In turn, each table consists of fields, organized into one or more columns by zero or more rows. All of the fields of a given column share the same attributes. See also: data base, field, row, table.

Command (in ITEQ) - Any operation that is processed by the ITEQ program, as distinguished from a TEQUEL statement, which is sent to the DBC/1012 for processing.

Configuration (in hardware) - The set of processor modules, disk storage units, cabinets, consoles, etc., connected by the same Ynet(s) that constitutes the make-up of a particular customer's DBC/1012 system.

Configuration (software component) - A program, running on the DBC/1012 under the control of the system console, that permits entering information about changes in the numbers, types, and relationships of processor modules, AMP clusters, and hosts in a DBC/1012 hardware configuration. See also reconfiguration.

Configuration Map (data structure) - A primitive file that defines the processor modules within a DBC/1012 configuration to the DBC/1012 system software.

Continue Request (in host software) - A message from an application program to the DBC/1012 to request that the DBC/1012 return more data from the same spool file.

Control Break (in TEQUEL) - The special row that contains summary information for the set of rows in a group, as specified by a FOR EACH clause in a RETRIEVE statement. Typically, the control break contains information generated by aggretate operators such as SUM and COUNT, and represents totals, subtotals, etc.

CREATE (in TEQUEL) - Any of the data definition statements that cause a new data structure (data base, user, table, view, index, or macro) to come into existence. Also, the process of creating new structures and updating the Data Dictionary/Directory to reflect their properties.

Creator (in TEQUEL) - The user who issued the CREATE statement for a structure within a data base. Typically, the creator has all privileges related to the created structure and controls dissemination of these privileges to other users.

Current Configuration (in the configuration program) - The configuration map that defines the hardware configuration on which the on-line software is currently executing. There may be hardware present in a DBC/1012 configuration that is not part of the current configuration.

Data Base (in TEQUEL) - A related set of tables that share a common space allocation and owner.

Data Base Computer (hardware) - A data base computing system, such as the DBC/1012.

Data Base Manager (software component) - A program, executed in each Access Module Processor), that satisfies requests for information from or changes to the data base. These requests are usually conveyed from the TEQUEL Parser or the Dispatcher via step messages.

Data Base Management System (generic term) - Computer procedures that permit the data base to be maintained independently of application programs. A data base management system provides services for data definition, data manipulation, and data integrity.

Data Base Recovery (process) - After a restart, any transactions that were in process at the restart are backed out. Also, if an Access Module Processor was off-line before the restart and is on-line after the restart, any tables with fallback are brought into synchronization with the other Access Module Processors in the cluster. These processes are completed before the DBC/1012 accepts new requests from the host.

Data Definition (in TEQUEL) - The statements and facilities that manipulate (CREATE, MODIFY, DROP, GRANT, REVOKE, GIVE, etc.) data base structures and the dictionary information kept about those structures.

Data Dictionary/Directory (in TEQUEL) - The information automatically maintained by the DBC/1012 about all of the data bases, tables, views, macros, and users known to DBC/1012 systems, including information about ownership, space allocation, accounting, and access right relationships between those objects.

Data Manipulation (in TEQUEL) - The statements and facilities that manipulate or change the information content of the data base. These statements include RETRIEVE, INSERT, UPDATE, and DELETE.

Data-Generating Statement (in ITEQ) - A TEQUEL statement that can potentially generate many result rows, causing the DBC/1012 to create a spool file. These statements include RETRIEVE, macro executions, and multi-statement TEQUEL requests.

DDL (in TEQUEL) - Data Definition (language).

ddname (in ITEQ) - Data definition name. In IBM operating systems, an entity that associates a logical file accessible to an application program with a physical data set, peripheral device, or subsystem service. In TSO, the connection between a ddname and a physical file (data set) or device is made by the allocate command.

Deadlock (in TEQUEL) - A condition in which two or more transactions are competing for locks on the same resources in such a way that none of the deadlocked transactions can make may progress without resources held by another transaction.

Default (in TEQUEL) - 1) A value for a field that will be supplied automatically by the DBC/1012 when a row is INSERTed, unless a value is explicitly entered by the user. 2) An attribute of a column that gives the default value.

Disk Storage Unit (in hardware) - A high-capacity secondary memory that is the principal medium for storage of information kept within a DBC/1012 system. The DBC/1012 provides one or two disks per Access Module Processor. Abbreviated DSU.

Diskette (in hardware) - A removable information storage medium that is used with the console. Diskettes can store the console processor's program, diagnostic programs for execution in the DBC/1012, images of the console display, and "scripts" of commands that can be sent from the console to the DBC/1012.

Dispatcher (software component) - A program that executes in each Interface Processor to coordinate the flow of steps from the TEQUEL Parser and sends them one at a time to the Data Base Manager in the Access Module Processor(s). It is also responsible for coordinating the return of information from the DBC/1012 to the application programs executing on the host.

DML (in TEQUEL) - Data Manipulation (Language).

DROP (in TEQUEL) - Any of the TEQUEL statements that remove a structure from the DBC/1012. The data contents of the dropped structure will also be removed from the DBC/1012, and the Data Dictionary/Directory will be adjusted appropriately.

DSU (in hardware) - Disk Storage Unit.

Dump (operation) - 1) A function provided by the Dump and Restore Utility to create an archival copy (typically on tape) of a data base, part of a data base, or a collection of data bases stored on the DBC/1012. Cf: Restore. 2) A process during DBC/1012 system restart that saves the contents of memory of each processor module for subsequent analysis to determine the cause of the restart. Used primarily to isolate software bugs.

Dump and Restore Utility (software component) - A program provided by Teradata that executes on the host computer to provide for the archiving of information from the DBC/1012 to tape and to restore archived information to the DBC/1012.

End User (in Data Dictionary/Directory) - An ordinary user of th DBC/1012, as opposed to a supervisory user or an administrator. An end-user cannot create a subordinate user or data base.

Exclusive Lock (in TEQUEL) - A kind of lock that precludes any other lock from being placed on the object. The DBC/1012 uses exclusive locks during data definition operations and utility restore operations.

Failure (in TEQUEL) - 1) Any condition that precludes complete processing of a TEQUEL statement. 2) The parcel in which the reason why processing could not be completed is reported to the user. Any failure will abort the current transaction.

Failback (in TEQUEL) - The ability of the DBC/1012 to maintain an extra copy of the every row of a table, so that if one or more AMPs are down (but not more than one AMP per cluster), all requests against the table can still be satisfied.

Fault (in hardware) - A physical condition that causes a device, component, or element to fail to perform in a required manner. For example, a short circuit, broken wire, or intermittent connection.

Field (in TEQUEL) - The basic unit of information stored in the DBC/1012. A field is either NULL or else has a single numeric or string value. See also column, row, table, data base.

Field Mode (in host software) - The mode of communication between an application program executing on a host computer and the DBC/1012 in which the DBC/1012 returns information converted for direct display and formatted for convenient manipulation of individual fields.

FILE (in ITEQ) - A command that places the result of the last data-generating statement into an MVS data set for subsequent manipulation by customer-written programs.

File (in Relational Data Base literature) - A term sometimes used to mean table.

File (in host software) - An imprecise term usually taken to mean an MVS data set.

Flow Control (software component) - Logic that regulates the rate at which the DBC/1012 accepts requests for work, so that internal overloads do not occur.

FORMAT (in TEQUEL) - A TEQUEL phrase that describes how the value in a field is to appear when returned to the host computer, particularly in field mode.

FORMAT (in ITEQ) - An ITEQ command that sets format mode.

FORMAT (in hardware) - The operation of analyzing the recording surfaces of a disk for flaws and writing the control information that prepares a disk for use by software. One of the commands provided by the Disk Maintenance Utility.

Format Mode (in ITEQ) - A mode in which results from data-generating statement(s) are presented in a manner suited to display or printing of reports.

Group (in TEQUEL) - A set of rows for which an aggregate operation, such as SUM, COUNT, MIN, or MAX, will be performed, as specified by the FOR EACH clause of a RETRIEVE statement. The HAVING clause specifies whether or not the aggregate result(s) for a particular group are to be output by the DBC/1012.

Hash Bucket (in hardware) - The hashing scheme or algorithm of the DBC/1012 computes a number called a hash bucket number from the key field of the record.

Hash Key (in hardware) - The field from which the hash bucket number is computed by the hashing algorithm.

Hash Map (in hardware) - A table in the Ynet Interface that specifies the allocation of hash bucket numbers to AMPS.

Hashing (in TEQUEL) - A way of mapping data records to various physical storage areas. In the DBC/1012, hashing determines on which AMP a given row is to be stored. The DBC/1012 hashing scheme operates in conjunction with the Ynet.

Hierarchical (generic term) - An organization of entities, such as data records, in which some "superior" or "parent" entities are related to one or more "subordinate" or "child" entities; also pertains to any data base management system that uses or describes information in a hierarchical form, such as IMS/VS. Cf: inverted, network, relational.

Host (generic term) - A general-purpose computer, attached to a DBC/1012, that can execute application programs that access and manipulate information on the DBC/1012.

Host System Interface (software component) - The interface that provides the means by which user- and Teradata-written application programs communicate with the DBC/1012. The Host System Interface consists of the Teradata Director Program, Call-Level Interface, and JES Subsystem Interface. Abbreviated HSI.

Host-Resident (generic term) - Of or pertaining to a software subsystem or application program that executes on a host computer.

HSRAM (in hardware) - High Speed Random Access Memory. A part of the Ynet Interface that stores packets to be transmitted on or received from the Ynet.

IFP (in hardware) - Interface Processor.

IFP Module (in hardware) - A complete IFP processor module, consisting of an IFP board, memory board, two Ynet Interface boards, and an IFP cable adapter board, and associated cables, etc.

Index (in TEQUEL) - A means of ordering and locating rows on disk for efficient access and processing. CF: primary index, secondary index, unique.

Interface Processor (in hardware) - A processor module in a DBC/1012 system that is primarily responsible for managing communication between the DBC/1012 and a host, and for transmitting users' requests into internal representations for processing by Access Module Processors. Abbreviated IFP. See also IFP module.

Inverted (generic term) - A form of organization of records in a data base management system in which extensive use is made of secondary index capability to provide alternative access paths to records. Each secondary index is also known as an "inversion." Cf: hierarchical, network, relational.

ITEQ (software component) - Interactive TEQUEL. A Teradata-supplied host-resident application program that provides a user at a 3270-series terminal the ability to directly use all of the query, data manipulation, and data definition capabilities of TEQUEL without any additional programming. ITEQ includes extensive features for the presentation of data, control of the 3270 screen, and generation of reports.

JCL (in COBOL Preprocessor) - Job Control Language: in the MVS operating system, the means by which execution of programs is requested and the relationship between a ddname and an MVS dataset or peripheral device is established.

Join (in TEQUEL) - A retrieval operation that combines information from two or more tables to produce a result. Cf: self-join.

Key (in TEQUEL) - The value of the index field(s) that are used to locate a row within the DBC/1012. Cf: hash key.

Keyword (in TEQUEL) - A string of characters that has a special meaning in the TEQUEL language. A keyword cannot be used as a name.

Lock (in TEQUEL) - The right to use a data base, table, or row for a particular purpose (such as to read or to write) with the assurance that other activities in the system cannot alter the object in a way that could affect the outcome of the activity that holds the lock.

Logical Host ID (in hardware) - An identifier assigned to a host computer duuring the configuration procedure, so that several host computers can be distinguished from one another.

Logical Processor ID (in the Configuration program) - A number that identifies a processor module within a DBC/1012 configuration. Cf: physical processor ID.

Macro (in TEQUEL) - A set of TEQUEL statements that are stored in the DBC/1012 and that can be executed by a single EXECUTE statement. A macro can have parameters that provide values to be substituted into the macro statement text as constants. Each macro execution is implicitly treated as a transaction.

Memory Management Unit (in hardware) - A part of the processor section of the AMP board and IFP board that provides a memory segmentation, dynamic relocation, and protection mechanism to software running on DBC/1012 processor modules. Abbreviated MMU.

Message (in host software) - The basic unit of information interchange between an application program and the DBC/1012. Messages consist of one or more parcels, which are logical subdivisions of a message. Messages are sent in one or more packets on the channel between the Teradata Director Program and the IFP.

Modifier (in TEQUEL) - A phrase that alters the default manner in which a TEQUEL statement is interpreted by the DBC/1012.

MODIFY (in TEQUEL) - Any of the data definition statements that dynamically alter the properties of a data base, macro, table, view, or user.

MVS (software component) - Multiple Virtual Storage: one of the primary operating systems (or "control programs") for medium and large IBM computers.

Name (in TEQUEL) - A word supplied by the user to refer to an object, such as a column, table, view, macro, user or data base.

Network (generic term) - A method or organizing records in a data base management system in which relationships between one record and another are represented by pointers. A pointer is the part of the record that gives the address, typically on disk, at which the next related record can be found. The data base thus consists of a network of records and pointers. This organizational form is also known as a "plex" structure, a "navigational" data base, or a "CODASYL model" data base. Cf: hierarchical, inverted, relational.

New Configuration (in the Configuration program) - Of or pertaining to the desired configuration of the DBC/1012 system after the reconfiguration process completes, as contrasted with the current configuration.

Node Logic (in hardware) - The electronics that implement the "tournament sort" tree of the Ynet.

Operator (in TEQUEL) - A symbol or keyword that specifies an operation to be performed on the values of the operands (if any).

Owner (in TEQUEL) - The owner of a data base is anyone above that data base in the hierarchy. An owner has the ability to GRANT or REVOKE all access rights to and from other users on any data base he owns. By default, the creator of the data base is the owner, but ownership can be transferred from one user to another by the GIVE statement. The immediate owner of a data base is the data base immediately above another data base in the hierarchy. The immediate owner of a table, view, or macro is the data base in which the table, view, or macro resides. The immediate owner is often required to have an appropriate privilege to execute a TEQUEL statement.

Packet (in host software) - The smallest unit of data sent on the channel between TDP and the IFP. A message consists of one or more packets. A packet is a purely physical division of a message. Packets are distinct from parcels, which are logical subdivisions of a message.

Parameter (in TEQUEL) - A variable name in a macro for which an argument value is substituted when the macro is executed. Also known as "formal parameter."

Parcel (in host software) - A logical part of a message. A parcel indicates what kind of information (TEQUEL statements, result rows, failure codes, etc.) it contains.

Parser (software component) - See TEQUEL Parser.

Password (in TEQUEL) - A string that must be entered when a user performs a LOGON to authenticate the identity of the user.

Physical Processor ID (in hardware) - The physical location of a processor module, consisting of the cabinet-ID and the backplane slot number.

Precedence (in TEQUEL) - TEQUEL executes operators in order of rank, rather than simply left to right. Precedence can be altered by parentheses.

Preprocessor Statement (in COBOL [PL/1] Preprocessor) - A statement that is processed by the COBOL [PL/1] Preprocessor and converted to calls on run-time library routine. COBOL [PL/1] Preprocessor statements are different from COBOL [PL/1] statements, which are passed on to the COBOL [PL/1] compiler unaltered, and TEQUEL statements, which are sent to the DBC/1012 for processing during the execution of a load module.

Primary Index (in TEQUEL) - An index used to determine on which AMP a table row is stored. It is also the means of locating a row on the AMP's disk. A primary index must be defined when a table is created.

Prime Key (in TEQUEL) - 1) A value in a primary index. 2) Of or pertaining to an operation that uses a primary index.

Primitive Directory (in the AMP) - A special primitive file that describes the locations of all other primitive files.

Primitive File (in the AMP) - A file on disk, used by the DBC/1012 during restart processing, that contains information in a format other than that of ordinary DBC/1012 data bases.

Privilege (in TEQUEL) - The right of a specified user to perform a specified TEQUEL statement (such as CREATE, RETRIEVE, GRANT, etc.) against a specified table, data base, user, macro, or view.

Processor Cluster (in hardware) - AMP cluster.

Processor ID (in hardware) - Physical processor ID.

Processor Module (in hardware) - See AMP module, IFP module.

PWBA (in hardware) - Printed Wiring Board Assembly.

Query (in TEQUEL) - 1) A start request. 2) The TEQUEL RETRIEVE statement.

Quiescent (system state) - A state of the DBC/1012 system that is on-line, but with no user sessions in process.

Read Lock (in TEQUEL) - A lock that indicates an intention to read but not alter data. Other users may simultaneously hold read locks on the same row or table, but a user requesting a write lock must wait until all outstanding read locks are released.

Reconfiguration (software component) - The program that redistributes data from AMPs in the current configuration to the processors that are added in the new configuration. See also Configurtion.

Record (generic term) - A set of related fields. Also, a term used loosely to mean row.

Record Mode (in host software) - The mode of communication between an application program executing on a host computer and the DBC/1012 in which the DBC/1012 returns information in packed data structures and host-oriented data representations intended for further processing by the application program. Cf: field mode.

Recovery (process) - See backout and data base recovery.

Recovery (software component) - The programs, executed as a part of restart processing, that start system recovery and data base recovery.

Relation (in relational data base terminology) - Synonym for table.

Relational (generic term) - A data base management system in which complex data structures are represented as simple, two-dimensional tables of columns and rows. Cf: hierarchical, inverted, network.

Request (in host software) - A message sent from an application program to the DBC/1012. Cf: response.

Reserved Word (in TEQUEL) - Synonym for keyword.

Resource Control Block (in host software) - A control block used by the Call-Level Interface to coordinate its processing. Abbreviated RCB.

Response (in host software) - A message sent from the DBC/1012 to an application program as a direct consequence of a request. The request-response protocol is half-duplex.

Response (in TEQUEL) - The result (success or failure) generated when the DBC/1012 processes a TEQUEL statement.

Restart (process) - The process by which on-line operation of the DBC/1012 is resumed following a system error condition such as a hardware failure, a software protocol failure, or loss and restoration of AC power.

Restore (software component) - A function provided by the Dump and Restore Utility that recreates a data base on the DBC/1012 from archived dump tapes. Cf: dump.

Restriction (in TEQUEL) - A sub-operation specified by a WHERE clause that specifies which rows of a table particpate in the data manipulation operation.

Result (in TEQUEL) - The information returned to the user to satisfy a request made of the DBC/1012. Results may include a return code, activity count, error message, warning message, title information, and/or rows from a spool file.

Row (in TEQUEL) - The fields, whether null or not, that represent one entry under each column in a table. The row is the smallest unit of information operated on by data manipulation statements. Cf: field, column, table, data base.

Secondary Index (in TEQUEL) - An index on a column or group of columns other than those used for the primary index. A secondary index (which results in storage of extra information ordered on the secondary index columns) can be used to more rapidly locate information in the DBC/1012.

Selection Map (in hardware) - Synonym for hash map.

Self-Join (in TEQUEL) - An operation in which a table is joined with itself to satisfy a query.

Separator (in TEQUEL) - A character or group of characters that separates words and special symbols in TEQUEL. Blanks and comments are the most common separators.

Session (in host software) - A logical connection between an application program on a host and the DBC/1012 that permits the application program to send one request to and receive one response from the DBC/1012 at a time.

Session Control (software component) - The software resident in the IFP that establishes, manages, recovers, and terminates sessions.

Spool File (in TEQUEL) - A file on the DBC/1012 that holds the result of TEQUEL statement processing until it can be examined by the user or application program.

Statement (in COBOL [PL/1] Preprocessor) - A preprocessor statement, TEQUEL statement, COBOL [PL-1] statement.

Statement (in ITEQ) - A TEQUEL statement (as opposed to an ITEQ command).

Statement (in TEQUEL) - A request for processing on the DBC/1012, consisting of a keyword verb and optional phrases and operands, that is processed as a single entity.

Step (in TEQUEL) - A unit of work that does some or all of the processing of a single TEQUEL statement. A step is created by the TEQUEL Parser and sent to the AMPs by the Dispatcher. Steps for a given statement are processed serially. Processing of a step is not initiated by the Dispatcher until the previous step has been completed by all affected AMPs.

Stream (in COBOL [PL/1] Preprocessor) - A named collection of records stored in the DBC/1012 as a spool file. The records are processed one at a time by the COBOL [PL/1] program.

Structure (in TEQUEL) - A table or data base.

Subquery (in TEQUEL) - Synonym for embedded RETRIEVE.

Supervisor Call (in host software) - A means of invoking a service of the MVS operating system by use of the SVC hardware instruction.

Supervisory User (in Data Dictionary/Directory) - A user who has been delegated authority by the administrator to further allocate DBC/1012 resources such as space allocation and the ability to CREATE, DROP, and MODIFY users within a subset of the overall user community.

SVC (in host software) - SuperVisor Call.

System View (in Data Dictionary/Directory) - A view that permits ordinary users, supervisory users, and administrators to obtain appropriate information about tables, views, macros, data bases, users, and the relationships among them.

Table (in TEQUEL) - A set of two or more columns by zero or more rows of fields of related information. See also data base.

Table Rebuild (software component) - A program that reconstructs on an AMP a portion of the DBC/1012 data base that was lost because of failure of a disk storage unit. Table rebuild uses the fallback copy of each row that is needed to reconstruct the data.

Table Recovery (process) - Synonym for data base recovery.

Task (generic term) - One of the threads of execution (or flows of control) in a multitasking environment.

TDP (software component) - Teradata Director Program.

tdpid (in host software) - An identifier that distinguisnes among several Teradata Director Program address spaces. Typically, the tdpid is allowed to default to the installation's standard production version of TDP.

TEQUEL Parser (software component) - The TEQUEL language processor, resident in the IFP, that translates TEQUEL statements entered by a user into the steps that accomplish the user's intentions.

TEQUEL Statement - A statement in the TEQUEL language that is processed by the DBC/1012.

Tera (generic term) - A prefix that means "trillion" (1,000,000,000,000).

Teradata Director Program (software component) - A program that manages communication between application programs and the DBC/1012. The Teradata Director Program is part of the Host System Interface. Abbreviated TDP.

Teradata Operating System (software component) - The control program that executes in every on-line DBC/1012 processor module. It can also run in an off-line processor module to support certain utility and diagnostic programs. Abbreviated TOS.

TOS (software component) - Teradata Operating System.

Transaction (in TEQUEL) - One or more TEQUEL statements that are processed as unit. Either all of the statements are executed normally, or any changes made during the transaction are backed out and the remainder of the statements in the transaction are not executed. If the transaction consists of more than one TEQUEL statement, it must start with a BEGIN TRANSACTION statement and end with an END TRANSACTION statement.

TSO (in ITEQ) - Time Sharing Option. A multi-user monitor subsystem that runs under the MVS operating system.

Tuple (in relational data base) - Synonym for row.

Type (in TEQUEL) - An attribute of a column that specifies the representation of data values for fields in that column. TEQUEL data types include numerics and strings.

Unique (in TEQUEL) - A property of an index that specifies that no two rows of a table are allowed to have the same key value for that index. The default is non-unique, which permits duplicate key values.

User (in TEQUEL) - 1) A person who uses the DBC/1012. 2) A data base, associated with that person, which stores that person's information and gives that person access to other DBC/1012 data bases.

View (in TEQUEL) - An alternate way of organizing and presenting information in the DBC/1012. A view, like a table, has rows and columns. However, the rows and columns of a view are not directly stored in the DBC/1012, but are derived from the rows and columns of tables (or other views) whenever the view is referenced.

Word (in TEQUEL) - One or more contiguous, nonblank alphabetic, numeric, or "national" characters ($, _, #). A name consists of one or more words, up to a total of 31 characters, counting each separator as one character.

Worker Task (software component) - A task in the AMP that executes the Data Base Manager program. There are many such worker tasks per AMP.

Write Lock (in TEQUEL) - A lock that indicates the intention to change information. If there is a write lock on an object, no other kind of lock can be allowed at the same time (except for the access lock). Cf: exclusive lock, read lock.

Ynet (in hardware) - The interconnection network that provides for high speed communication among the processor modules of a DBC/1012 system.

APPENDIX C
TEQUEL LANGUAGE DESCRIPTIONS

The facilities of the TEQUEL language are described in this Appendix. The language has been partitioned into three general classes of statements: Data Definition (DDL), Data Manipulation (DML), and Data Administration (DAL).

C.1 DDL Statement Descriptions

C.1.1 CREATE TABLE

Defines the format and characteristics of a table. The parameters are:

- Table Name
- Fallback or no Fallback (Fallback is default)
- Column definitions:
  - Column Name
  - Data Type
    - Integer (-4*10^15 to 4*10^15)
    - Decimal (precision=1 to 15, scale=1 to n)
    - Floating (+ or - 4*10^-307 to + or - 2*10^308)
    - Fixed length text (max. 30,000 chs.)
    - Variable length text (max. 30,000 chs.)
    - Fixed length binary string (max. 30,000 bytes)
    - Var. length binary string (max. 30,000 bytes)
    - Date
    - Null option
      - NULL (null allowed, system null value used)
      - NO NULL (null not allowed)
      - DEFAULT with value (specified value used)
    - Display format for reports
    - Report column header specification

Primary Index specification (One required, max. 16 columns, unique or not unique. Unique is default). Secondary Index specification (Up to 16. Other characteristics same as primary index).

C.1.2 CREATE INDEX

Used to add indexes to an already existing table definition. Also causes an index subtable to be created for subsequent use during DML operations. The parameters are:

Unique or not unique (Unique is default)
Names of columns in index (max. 16)
Name of table for which index is being created

Note that there is no provision to name the index.

C.1.3 DROP TABLE

Deletes a table definition from the dictionary/directory. A table must be empty of data before it can be deleted. Parameters are:

Name of the table definition to be deleted

C.1.4 DROP INDEX

Deletes an index definition from the data/dictionary. Parameters are:

Name(s) of the columns constituting the index
Name of the table the index applies to

Note again that there is no provision for using an index name. To DROP an index requires that all columns defined for the index must be individually listed. If all columns participating in the index are not specified in the DROP statement, an error message will be returned.

C.1.5 MODIFY TABLE

Used to modify an existing table definition by either adding or deleting columns. Parameters are:

Name of the table to be modified
Name(s) of the columns involved in the modification
For each column, either a DROP command (to delete a column)
or a description of the column (to add a column)

To modify the data description of a column in a table requires first DROPing the column, then entering the new description. Changing the name of the table and adding new secondary indexes are provided for by other commands (CREATE INDEX and RENAME TABLE). To change the primary index or to change the fallback option requires that a new table be defined, followed by a series of DML statements to move the data from the old table to the new table, followed by deletion of the old table (DELETE ALL, then DROP TABLE).

C.1.6 RENAME TABLE

Changes the name of an existing table. Parameters are:

Old table name
New table name

C.2 DML Statement Descriptions

C.2.1 DELETE

Used to delete rows from a table. Parameters are:

Table name
Conditional clause (described under RETRIEVE)
or
ALL

3.2.2 INSERT

Used to add new rows to a table. Parameters are:

Table name
Constants representing data values to be inserted in columns in order of original definition
or
Column name/constant pairs, in arbitrary order

An "= RETRIEVE statement" clause is included as another option for this command. It is used to locate a row from another table which is to be appended to the table being updated.

C.2.3 RETRIEVE

Used to retrieve rows from a table or tables. Supports SELECT, PROJECT and JOIN operations via context and clauses. Also can re-order position of columns, retrieve in sorted order, retrieve columns based upon aggregate operations (e.g., list names of employees in all departments having total salaries greater than $1M), change the name of columns and

return rows by group (e.g., employees by department). RETRIEVE may be nested. Parameters are:

- Specification of whether retrieval is on unique key
- Table name/column name pairs to be delivered
- Conditional clause
  - Comparison operators
    - EQ, GT, LT, NE, LE, GE, BETWEEN...AND
  - Logical operators
    - AND, OR, NOT
  - Partial string matches
    - BEGINS WITH
    - ENDS WITH
    - CONTAINS
    - negatives of the above
  - Set operators
    - IN or NOT IN a set of values
      - Set is a list of constants
      - or
      - Set is the result of a RETRIEVE
  - Aggregate operators
    - AVERAGE
    - COUNT
    - MAXIMUM/MINIMUM
    - SUM
- Specification of sort order
  - Column name for sorting
  - DESCending or ASCending
- Arithmetic operators
  - +, -, *, /, MOD(ulo)
  - (may also be used in the conditional clause)
- Aggregate operators
  - AVERAGE, COUNT, MAXIMUM, MINIMUM, SUM
  - (may also be used in the conditional clause)

ALIASes for columns
Control break specification
Name of column(s) to deliver
Name of column(s) to perform break on

C.2.6 UPDATE

Used to change values in existing rows of a table. Parameters are:

Table name
Column name/column value pairs (value can be constant or referenced)
Conditional clause (described under RETRIEVE)

C.3 DAL Statement Descriptions

C.3.1 CREATE DATABASE

Used to allocate system storage resources to application areas or organizations. Parameters are:

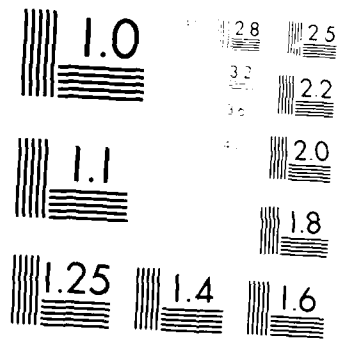
Name to be given to new data base
Name of owner data base from which resources will come
Number of bytes to be allocated for the new data base
Number of bytes allocated for the spool file
Fallback or no fallback option (fallback is default)
Account ID for billing of system resource usage

C.3.2 CREATE USER

Used to allocate system storage resources to individual users. This statement is the same as the CREATE DATABASE statement, with the addition of two more parameters.

Password
A sequence of startup TEQUEL statements which set the environment for the user (e.g., default settings)

AD-A169 300 C3I (COMMAND CONTROL COMMUNICATIONS AND INTELLIGENCE) 3/3
TERADATA STUDY(U) MEASUREMENT CONCEPT CORP ROME NY
J DECKER MAR 86 RADC-TR-85-273 F30602-85-C-0029
UNCLASSIFIED F/G 17/2 NL


1.0
2.8
2.5
2.2
1.1
2.0
1.8
1.25
1.4
1.6

C.3.3 DROP DATABASE and DROP USER

Used to delete data bases or users from the system. There is one parameter:

Name of data base or user to be deleted

All tables and users within a data base or user space must be dropped before a DROP statement may be entered for a data base or users.

C.3.4 GRANT

Used to grant data base access privileges to users. Parameters are:

A list of privileges to be granted, or ALL
Name of data base or data base/table upon which privileges are being granted
Name of user getting privileges, or ALL
or
Name of organization and ALL (users)

C.3.5 MODIFY DATABASE and MODIFY USER

Used to change a data base or user definition. Any of the parameters used in the CREATE DATABASE and CREATE USER statements may be changed.

C.3.6 REVOKE

Used to revoke privileges. Same parameters as the GRANT statement.

C.4 Miscellaneous Constructs

These statements, clauses and modifiers apply to two or more of the three classes (DDL, DML, DAL).

C.4.1 BEGIN TRANSACTION and END TRANSACTION

Used to bracket a series of TEQUEL statements in support of transaction processing. Parameters are:

TEQUEL statements

C.4.2 CREATE MACRO

Used to define and save a series of parameterized TEQUEL statements for repeated invokation by programs or interactive users. Only constants may be used as parameters. Parameters are:

Parameter definitions:
- Parameter name
- Parameter description
  - Number of CHARACTERS or DIGITS
  - Default value
- TEQUEL statements

C.4.3 CREATE VIEW

This statement is used to establish a tailored subset of the data base for subsequent use by other TEQUEL statements. This subset is, in effect, a non-stored relation. It can define the participating rows via conditional clauses and can define the participating columns by naming them. The columns can come from more than one table. The relational operators (SELECT, PROJECT and JOIN) are all embodied in this one construct. Views thus defined can be further SELECTed and PROJECTed when used by other TEQUEL statements, such as RETRIEVE. In addition, columns can be

reordered and renamed, and "virtual" columns can be defined to contain the results of computations on other columns. Parameters are:

VIEW name
Table name/column name pairs defining source columns
New names for columns projected from existing tables
Names for virtual columns
Data descriptions for virtual columns (no information as to format)
Conditional clause for selection of rows

C.4.4 EXECUTE

Causes the execution of a macro. Parameters are:

Macro name
Macro parameter values
or
Macro parameter name/parameter value pairs

C.4.5 LOCKING

Used as a modifier for DML statements to lock out other users from data until transaction is completed. Precedes DML statement for which locks are invoked. Parameters are:

Data base or data base/table name to be locked
Specification of locks to be invoked
    READ, WRITE, EXCLUSIVE, ACCESS

C.4.6 USING

Describes data expected from the host. Values in the host data record, which must be submitted in the host request along with the TEQUEL statement, correspond to the description of values in the USING modifier.

Functionality is similar to argument string passing. Allows TEQUEL statement using the argument string to be cached. Parameters are: Name/data description pair.

C.4.7 ABORT

Used to abort a TEQUEL statement or macro if a specified condition occurs. If ABORT is executed, locks on the data base are released, changes to data are backed out, and spooled output is deleted. Parameters are:

Message
Conditional clause

C.4.8 COMMENT

Used to enter comments on a table, column, view, or macro, or to retrieve comments previously entered. Comments must be enclosed in apostrophes or quotation marks and may be no longer than 255 characters in length. Parameters are:

Name of table, column, view or macro
Comment text

C.4.9 DATABASE

Used to set a default data base name so that it need not be entered in each TEQUEL statement. Parameters are:

Name of data base

C.4.10 DROP VIEW and DROP MACRO

Deletes a view or macro definition from the data/dictionary. Parameters are:

Name of the view or macro to be deleted

C.4.11 ECHO

Returns a fixed character string to the requestor. Used primarily in macros to return commands to ITEQ or BTEQ session. Parameters are:

A string or command

C.4.12 RENAME VIEW and RENAME MACRO

Used to change the name of a view or macro. Parameters are:

Old name of view or macro
New name of view or macro

3.4.13 REPLACE VIEW and REPLACE MACRO

Used to replace a view or macro definition with a new one. Parameters are:

Name of view or macro to be replaced
The new definition

APPENDIX D

MODEL RUN OUTPUTS

INTERACTIVE - PRIMARY KEY RETRIEVAL

PROCESSORS	TRANSACTION PER SECOND	AMP/IFP	BOTTLENECK
4	5.58	1.00	IFP
8	8.38	1.67	IFP
16	16.77	1.67	IFP
32	30.75	1.91	IFP
64	61.47	1.91	IFP
128	120.17	1.98	IFP
256	240.33	1.98	IFP
512	477.87	1.99	IFP
1024	955.33	1.99	IFP

INTERACTIVE - PRIMARY KEY RETRIEVAL - BALANCED

PROCESSORS	TRANSACTION PER SECOND	AMP/IFP	BOTTLENECK
4	5.58	1.00	IFP
8	11.18	1.00	IFP
16	22.35	1.00	IFP
32	44.73	1.00	IFP
64	89.47	1.00	IFP
128	178.93	1.00	IFP
256	357.87	1.00	IFP
512	715.73	1.00	IFP
1024	1431.48	1.00	IFP

****

BATCH - PRIMARY KEY RETRIEVALS

PROCESSORS	TRANSACTION PER SECOND	AMP/IFP	BOTTLENECK
4	19.47	1.00	IFP
8	29.20	1.67	IFP
16	58.40	1.67	IFP
32	107.10	1.91	IFP
64	214.20	1.91	IFP
128	418.67	1.98	IFP
256	837.35	1.98	IFP
512	1665.00	1.99	IFP
1024	3330.00	1.99	IFP

BATCH - PRIMARY KEY RETRIEVALS - BALANCED

PROCESSORS	TRANSACTION PER SECOND	AMP/IFP	BOTTLENECK
4	19.47	1.00	IFP
8	38.95	1.00	IFP
16	77.90	1.00	IFP
32	155.78	1.00	IFP
64	311.57	1.00	IFP
128	623.13	1.00	IFP
256	1246.28	1.00	IFP
512	2492.57	1.00	IFP
1024	4985.10	1.00	IFP

INTERACTIVE - SECONDARY KEY RETRIEVAL			*	INTERACTIVE - SECONDARY KEY RETRIEVAL - BALANCED		
PROCESSORS	TRANSACTION PER SECOND	AMP/IFP BOTTLENECK	*	PROCESSORS	TRANSACTION PER SECOND	AMP/IFP BOTTLENECK
4	2.77	1.00 IFP	*	4	2.77	1.00 IFP
8	4.15	1.67 IFP	*	8	5.38	1.00 DSU
16	6.82	1.67 DSU	*	16	6.80	1.67 IFP/DSU
32	7.40	1.91 DSU	*	32	7.47	3.57 DSU
64	7.70	1.91 DSU	*	64	7.75	7.00 DSU
128	11.72	1.98 DSU	*	128	11.78	7.53 DSU
256	11.87	1.98 DSU	*	256	11.93	12.47 DSU
512	11.97	1.99 DSU	*	512	12.00	15.00 DSU
1024	12.02	1.99 DSU	*	1024	12.03	15.00 DSU

BATCH - SECONDARY KEY RETRIEVALS			*	BATCH - SECONDARY KEY RETRIEVALS - BALANCED		
PROCESSORS	TRANSACTION PER SECOND	AMP/IFP BOTTLENECK	*	PROCESSORS	TRANSACTION PER SECOND	AMP/IFP BOTTLENECK
4	4.32	1.00 DSU	*	4	4.32	1.00 DSU
8	5.98	1.67 DSU	*	8	6.35	3.00 DSU
16	6.35	1.67 DSU	*	16	7.15	7.00 DSU
32	7.42	1.91 DSU	*	32	7.58	15.00 DSU
64	7.70	1.91 DSU	*	64	7.78	15.00 DSU
128	11.72	1.98 DSU	*	128	11.82	15.00 DSU
256	11.90	1.98 DSU	*	256	11.95	15.00 DSU
512	11.98	1.99 DSU	*	512	12.00	15.00 DSU
1024	12.03	1.99 DSU	*	1024	12.03	15.00 DSU

I/A - SECONDARY RETRIEEVAL - 2:1			*I/A - SECONDARY RETRIEVAL - BALANCED			
PROCESSORS	TRANSACTION PER SECOND	AMP/IFP BOTTLENECK	* PROCESSORS	TRANSACTION PER SECOND	AMP/IFP BOTTLENECK	
4	2.78	1.00 IFP	* 4	2.78	1.00 IFP	
8	4.17	1.67 IFP	* 8	5.57	1.00 IFP	
16	8.35	1.67 IFP	* 16	11.13	1.00 IFP	
32	15.30	1.91 IFP	* 32	22.27	1.00 IFP	
64	30.62	1.91 IFP	* 64	44.53	1.00 IFP	
128	59.88	1.98 IFP	* 128	89.12	1.00 IFP	
256	119.77	1.98 IFP	* 256	178.25	1.00 IFP	
512	238.15	1.99 IFP	* 512	356.52	1.00 IFP	
1024	476.32	1.99 IFP	* 1024	713.02	1.00 IFP	

BATCH - SECONDARY RETREVAL - AL 2:1			*BATCH - SECONDARY RETRIEVAL - BALANCED			
PROCESSORS	TRANSACTION PER SECOND	AMP/IFP BOTTLENECK	* PROCESSORS	TRANSACTION PER SECOND	AMP/IFP BOTTLENECK	
4	8.63	1.00 DSU	* 4	8.63	1.00 DSU	
8	28.88	1.67 IFP/DSU	* 8	28.88	1.67 IFP/DSU	
16	58.03	1.67 IFP	* 16	60.60	1.29 DSU	
32	106.47	1.91 IFP	* 32	131.22	1.29 DSU/IFP	
64	212.93	1.91 IFP	* 64	269.35	1.29 IFP/DSU	
128	416.33	1.98 IFP	* 128	619.38	1.00 IFP	
256	832.68	1.98 IFP	* 256	1238.85	1.00 IFP	
512	1655.68	1.99 IFP	* 512	2477.75	1.00 IFP	
1024	3311.33	1.99 IFP	* 1024	4955.67	1.00 IFP	

INTERACTIVE - UPDATE				*	INTERACTIVE - UPDATE - BALANCED			
PROCESSORS	TRANSACTION PER SECOND	AMP/IFP	BOTTLENECK	*	PROCESSORS	TRANSACTION PER SECOND	AMP/IFP	BOTTLENECK
4	1.43	1.00	DSU	*	4	1.43	1.00	DSU
8	2.58	1.67	DSU	*	8	4.28	3.00	DSU
16	7.15	1.67	IFP/DSU	*	16	8.55	3.00	DSU
32	15.02	1.91	DSU	*	32	17.10	3.00	IFP/DSU
64	30.10	1.91	DSU	*	64	34.63	3.27	IFP/DSU
128	89.45	1.98	DSU	*	128	92.28	2.28	IFP/DSU
256	178.90	1.98	DSU	*	256	184.57	2.28	IFP/DSU
512	358.85	1.99	DSU	*	512	369.13	2.28	IFP/DSU
1024	717.70	1.99	DSU	*	1024	738.28	2.27	IFP/DSU

INTERACTIVE - JOIN				*	INTERACTIVE - JOIN - BALANCED			
PROCESSORS	TRANSACTION PER SECOND	AMP/IFP	BOTTLENECK	*	PROCESSORS	TRANSACTION PER SECOND	AMP/IFP	BOTTLENECK
4	0.22	1.00	DSU	*	4	0.22	1.00	DSU
8	0.57	1.67	DSU	*	8	0.67	3.00	DSU
16	1.10	1.67	DSU	*	16	1.52	7.00	DSU
32	2.18	1.91	DSU	*	32	2.93	15.00	IFP/DSU
64	3.87	1.91	DSU	*	64	5.02	15.00	DSU
128	7.23	1.98	DSU	*	128	9.10	15.00	DSU
256	11.17	1.98	DSU	*	256	13.25	15.00	DSU
512	15.32	1.99	DSU	*	512	17.17	15.00	DSU
1024	18.90	1.99	DSU	*	1024	20.12	15.00	DSU

I/A - JOIN - 2:1				*I/A - JOIN - BALANCED			
PROCESSORS	TRANSACTION PER SECOND	AMP/IFP	BOTTLENECK	*PROCESSORS	TRANSACTION PER SECOND	AMP/IFP	BOTTLENECK
4	0.42	1.00	DSU	* 4	0.42	1.00	DSU
8	2.30	1.67	DSU	* 8	2.93	3.00	IFP
16	8.82	1.67	IFP	* 16	9.00	1.29	DSU
32	16.18	1.91	IFP	* 32	23.50	1.00	IFP
64	32.38	1.91	IFP	* 64	47.08	1.00	IFP
128	63.28	1.98	IFP	* 128	88.93	1.10	AMP/IFP
256	100.72	1.98	AMP	* 256	101.08	2.61	AMP/IFP
512	102.90	1.99	AMP	* 512	103.23	5.65	AMP
1024	103.62	1.99	AMP	* 1024	103.77	10.91	AMP

INTERACTIVE - PI MIX				*	INTERACTIVE - PI MIX - BALANCED			
PROCESSORS	TRANSACTION PER SECOND	AMP/IFP	BOTTLENECK	*	PROCESSORS	TRANSACTION PER SECOND	AMP/IFP	BOTTLENECK
4	4.18	1.00	DSU	*	4	4.18	1.00	DSU
8	8.35	1.67	DSU	*	8	7.52	1.00	DSU
16	12.78	1.67	IFP	*	16	13.11	1.29	DSU
32	19.80	1.91	DSU	*	32	19.15	2.56	DSU/AMP
64	34.55	1.91	DSU	*	64	25.44	7.57	DSU
128	41.21	1.98	DSU	*	128	42.25	4.57	DSU
256	44.46	1.99	DSU	*	256	45.28	8.14	DSU
512	46.38	1.99	DSU	*	512	46.78	12.47	DSU
1024	47.26	1.99	DSU	*	1024	47.52	15.79	DSU

INTERACTIVE - I&W MIX				*	INTERACTIVE - I&W MIX - BALANCED			
PROCESSORS	TRANSACTION PER SECOND	AMP/IFP	BOTTLENECK	*	PROCESSORS	TRANSACTION PER SECOND	AMP/IFP	BOTTLENECK
4	0.93	1.00	DSU	*	4	0.92	1.00	DSU
8	1.85	1.67	DSU	*	8	2.50	3.00	DSU
16	3.71	1.67	DSU	*	16	4.70	7.00	DSU
32	5.80	1.91	DSU	*	32	7.20	15.00	DSU
64	8.28	1.91	DSU	*	64	9.35	15.00	DSU
128	12.95	1.98	DSU	*	128	15.33	15.00	DSU
256	16.47	1.98	DSU	*	256	17.38	15.00	DSU
512	18.10	1.99	DSU	*	512	18.63	15.00	DSU
1024	19.35	1.99	DSU	*	1024	19.33	15.00	DSU

I/A - PI - 2:1				*	I/A - PI - BALANCED			
PROCESSORS	TRANSACTION PER SECOND	AMP/IFP	BOTTLENECK	*	PROCESSORS	TRANSACTION PER SECOND	AMP/IFP	BOTTLENECK
4	4.27	1.00	IFP	*	4	4.27	1.00	IFP
8	6.44	1.67	IFP	*	8	8.56	1.00	IFP
16	12.87	1.67	IFP	*	16	17.10	1.00	IFP
32	23.60	1.91	IFP	*	32	34.21	1.00	IFP
64	47.21	1.91	IFP	*	64	68.45	1.00	IFP
128	92.28	1.98	IFP	*	128	137.24	1.00	IFP
256	184.58	1.98	IFP	*	256	274.47	1.00	IFP
512	367.03	1.99	IFP	*	512	548.97	1.00	IFP
1024	734.07	1.99	IFP	*	1024	1097.85	1.00	IFP

I A - IW - 2:1				*	I/A - IW - BAL			
PROCESSORS	TRANSACTION PER SECOND	AMP/IFP	BOTTLENECK	*	PROCESSORS	TRANSACTION PER SECOND	AMP/IFP	BOTTLENECK
4	1.83	1.00	DSU	*	4	1.83	1.00	DSU
8	6.20	1.67	IFP	*	8	6.32	1.00	DSU
16	12.42	1.67	IFP	*	16	16.50	1.00	IFP
32	22.77	1.91	IFP	*	32	33.00	1.00	IFP
64	45.53	1.91	IFP	*	64	66.20	1.00	IFP
128	89.02	1.98	IFP	*	128	132.47	1.00	IFP
256	178.02	1.98	IFP	*	256	264.83	1.00	IFP
512	353.78	1.99	IFP	*	512	425.50	1.40	AMP/IFP
1024	473.75	1.99	AMP	*	1024	477.78	3.10	AMP IFP

BATCH - TERRA BENCH - 2:1			BOTTLENECK	*	BATCH - TERRA BENCH - BAL		
PROCESSORS	TRANSACTION PER SECOND	AMP/IFP		* PROCESSORS	TRANSACTION PER SECOND	AMP/IFP	BOTTLENECK
4	19.48	1.00 IFP		* 4	29.22	1.00 IFP	
32	107.18	1.91 IFP		* 32	155.87	1.00 IFP	
60	194.87	2.00 IFP		* 60	293.25	1.00 IFP	

BATCH - TERRA BENCH - 2:1 - EXTENDED			BOTTLENECK	*	BATCH - TERRA BENCH - BAL - EXTENDED		
PROCESSORS	TRANSACTION PER SECOND	AMP/IFP		* PROCESSORS	TRANSACTION PER SECOND	AMP/IFP	BOTTLENECK
4	19.48	1.00 IFP		* 4	19.48	1.00 IFP	
8	29.22	1.67 IFP		* 8	38.97	1.00 IFP	
16	53.47	1.67 IFP		* 16	77.93	1.00 IFP	
32	107.18	1.91 IFP		* 32	155.87	1.00 IFP	
64	214.25	1.91 IFP		* 64	311.73	1.00 IFP	
128	418.97	1.98 IFP		* 128	623.48	1.00 IFP	
256	837.92	1.98 IFP		* 256	1246.97	1.00 IFP	
512	1666.08	1.99 IFP		* 512	2493.92	1.00 IFP	
1024	3332.25	1.99 IFP		* 1024	4987.92	1.00 IFP	

BATCH - TERRA BENCH - 2:1 - MODIFIED			BOTTLENECK	*	BATCH - TERRA BENCH - BAL - MODIFIED		
PROCESSORS	TRANSACTION PER SECOND	AMP/IFP		* PROCESSORS	TRANSACTION PER SECOND	AMP/IFP	BOTTLENECK
4	4.07	1.00 DSU		* 4	4.07	1.00 DSU	
8	10.18	1.67 DSU		* 8	12.22	3.00 DSU	
16	20.37	1.67 DSU		* 16	26.43	4.33 DSU	
32	42.78	1.91 DSU		* 32	52.87	4.33 DSU	
64	85.53	1.91 DSU		* 64	106.05	4.82 DSU	
128	245.53	1.98 DSU		* 128	281.67	3.27 DSU	
256	491.08	1.98 DSU		* 256	563.50	3.34 DSU	
512	985.00	1.99 DSU		* 512	1127.50	3.30 DSU	
1024	1970.00	1.99 DSU		* 1024	2255.00	3.30 DSU	

APPENDIX E

COMPARISON OF TEQUEL AND IBM/SQL

The following pages, extracted from the DBC Data Base Computer Reference Manual [DBC-11], detail the lexical, syntax, and semantic differences between IBM SQL and TEQUEL.

2.3 COMPARISON OF IBM SQL AND TERADATA SQL

The RETRIEVE, UPDATE, INSERT, and DELETE operations permit equivalent IBM SQL keywords and SQL-like syntax so that a user who is familiar with IBM SQL can use these constructs instead of TEQUEL. Note that this SQL-like syntax is not an IBM SQL-compatible syntax (that is, the syntax is not completely compatible with IBM SQL). Note also that there is only one parser (the TEQUEL Parser) that has TEQUEL lexical rules, TEQUEL expression syntax, and TEQUEL Data Definiton statements. (There are no SQL-like syntax features for DDL statements). Hence, there is only one RETRIEVE (or SELECT), UPDATE, INSERT, or DELETE statement where the user can use SQL-like syntax or TEQUEL syntax, or intermix the two.

Some of the more noticeable differences between Teradata SQL-like syntax and IBM SQL follow.

2.3.1 Lexical Differences

These are the major lexical differences between Teradata SQL-like syntax and IBM SQL.

- In TEQUEL, names can have 31 characters, instead of 18 characters as in SQL.
- In TEQUEL, spaces are legal in names. In SQL, names with spaces must be enclosed by double quotation marks.
- In TEQUEL, double quotation marks can be string delimiters. In SQL, strings must be delimited by single quotation marks: double quotation marks are used only for names.
- Host variables in TEQUEL statements start with '@'; in SQL, they start with ':'.

2.3.2 Syntax Differences

These are the major syntactic differences between Teradata SQL-like syntax and IBM SQL.

- A common problem when translating SQL to TEQUEL occurs with the aggregate expression, "SUM x". The TEQUEL "SUM x" is equivalent to the SQL "SUM(x)". SUM(x) can be used in TEQUEL, but often gives an unexpected result. For example, the clause

  ```
  HAVING SUM(x) + SUM(y) > 10
  ```

  gives the expected result in SQL, but does not give the same result in TEQUEL. TEQUEL parses this clause as

  ```
  HAVING SUM((x) + SUM y) > 10
  ```

  which does not give the desired result. The equivalent clause in TEQUEL is

  ```
  HAVING SUM x + SUM y > 10
  ```

- The expression

  ```
  X > 5 AND Y > 10 OR Z > 20
  ```

  is legal in SQL, and means the same as

  ```
  (X > 5 AND Y > 10) OR Z > 20
  ```

  But in TEQUEL, the expression without the parentheses is illegal because TEQUEL requires the parentheses to avoid ambiguity.

- In SQL, a "." is used to qualify a table name. In TEQUEL a "." or ":" can be used. For example, in TEQUEL Database:Table or DataBase.Table are equivalent.

- In TEQUEL, the "FOR UPDATE OF" clause is not allowed in a query statement.

- In TEQUEL, the "WHERE CURRENT OF CURSOR" clause is not allowed in an UPDATE or DELETE statement.

- All of the DDL statements have different syntax between SQL and TEQUEL.

- SQL and TEQUEL have a completely different set of reserved words.

2.3.3 Semantic Differences

These are the semantic differences between TEQUEL and SQL. This list is not complete.

- NULL rules are different. NULL = NULL is true in TEQUEL, but NULL = NULL is null in SQL. SQL uses a tri-state Boolean logic, whereas TEQUEL uses a bi-state Boolean logic.
- SQL does not allow conversions between character and numeric data type. TEQUEL allows all conversions.
- TEQUEL does not support correlated subqueries. Because the EXISTS predicate relies on correlated subqueries, TEQUEL also does not support EXISTS.
- TEQUEL does not support UNION, INTERSECT, or MINUS.
- TEQUEL supports only a small subset of the LIKE clause.
- TEQUEL does not allow comparisons (>,<,>=,<=,=, =) to be used against subqueries.
- TEQUEL does not support ANY or ALL followed by a comparison.
- TEQUEL currently does not support COUNT/AVG/SUM(DISTINCT columnname).
- TEQUEL does not support UPDATE/DELETE WHERE CURRENT OF cursor, and does not have an equivalent to the cursor concept.
- In SQL, duplicate rows are allowed in a table. To select only one of the duplicate rows, the UPDATE/DELETE WHERE CURRENT OF cursor is used. TEQUEL does not allow duplicate rows.
- All of the DDL statements have different semantics. The most noticeable difference is that the TEQUEL CREATE TABLE statement requires that a primary index be specified, but in SQL all indexes are built after the table is created.
- All of the Data Dictionary tables are completely different.
- The preprocessor interface is completely different. Host programs that need to look at the preprocessor structures (SQLDA, SQLTIB, etc.) cannot be transported to use TEQUEL.

2.3.4 Examples of the SQL-Like Syntax

The following examples show the use of the SQL-like syntax in SELECT, INSERT, UPDATE, and DELETE statements.

```
SELECT * FROM CFM.TempL WHERE WorkDept = 'D11' ORDER BY 1;

SELECT DISTINCT LastName FROM TempL WHERE LastName = 'Brown';

SELECT EmpNo,FirstName,LastName,EducLvl FROM TempL
    WHERE CFM.TempL.EducLvl IN (16, 18, 20);

SELECT EmpNo,FirstName,LastName,WorkDept FROM TempL
    WHERE LastName LIKE 'P%' AND WorkDept LIKE '%D';

SELECT COUNT(*),AVG(Salary),MAX(Salary),
       MIN(Salary),SUM(Salary)
   FROM TempL WHERE WorkDept = 'D11';


INSERT INTO TDept VALUES ( 'D41',
          'Systems Test', '000190', 'D01' );

INSERT INTO CFM.TempL
    (EmpNo, FirstName, MidInit, LastName,
                WorkDept, HireDate, Sex)
    VALUES ( '000410', 'Howard', ' ',
            'Jensen', 'D41', 820308, 'M');

INSERT INTO EmpD41
    ( EmpNo, FirstName, MidInit,
      LastName, WorkDept, HireDate, Sex )
  SELECT EmpNo, FirstName,MidInit,
         LastName, WorkDept,HireDate,Sex
  FROM TempL
   WHERE EmpNo LIKE '0004%'
      Or EmpNo IN ( SELECT MgrNo FROM TDept
                          WHERE DeptNo = 'D41');

UPDATE TempL SET JobCode = 52,
           Salary = 18140, HireDate = NULL
   WHERE EmpNo = '000410' OR EmpNo IS NULL;


DELETE FROM TDept
    WHERE MgrNo IN ( SELECT EmpNo FROM EmpD41 );
```

The following examples show how TEQUEL and SQL syntax can be intermixed.

```
RET UNIQUE * FROM Employee ;
RET DISTINCT A FROM Employee WHERE X IS NULL AND Y = NULL;
SELECT X,Employee.Y WHERE A BEGINS WITH 'X';
SELECT Employee.Y FOR EACH X;
```

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control, Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of competence is provided to ESD Program Offices (POs) and other ESD elements to perform effective acquisition of C^3I systems. The areas of technical competence include communications, command and control, battle management, information processing, surveillance sensors, intelligence data collection and handling, solid state sciences, electromagnetics, and propagation, and electronic, maintainability, and compatibility.

END

DTIC

8-86